

# MacTech®

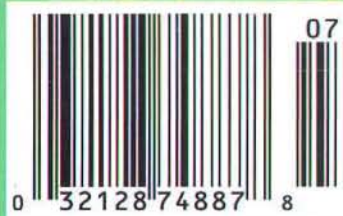
INSIDE  
PUZZLE:  
Mixed Up Threads

Includes Special *develop* → Section by Apple Computer, Inc.

## THE QUICKTIME MEDIA LAYER

## PROGRAMMING WITH QUICKTIME VR

**PLUS** OPENSTEP Sockets

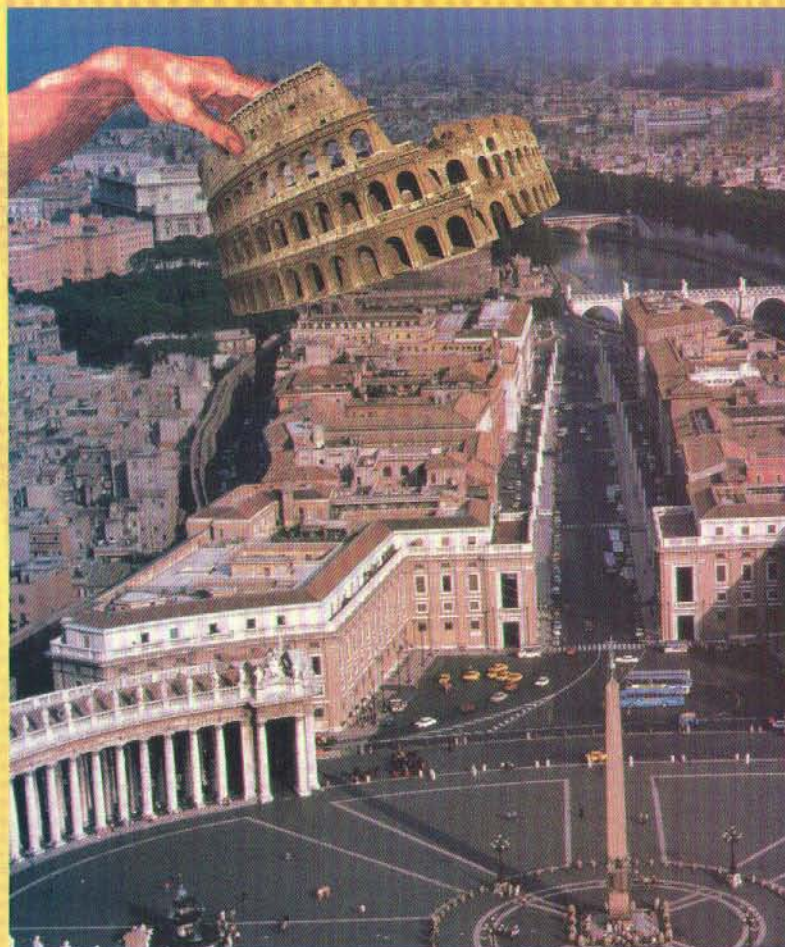


QuickTime and the QuickTime Logo are trademarks of Apple Computer, Inc.

\$5.85 US  
\$6.95 Canada  
ISSN 1067-8360  
Printed in U.S.A.



## Build Rome in a day. It's easy with Symantec Visual Café™ for Java™



### Visual Java Development



Develop faster. Compile faster. Debug faster. Realize your Java™ dreams faster.

Symantec, the creator of the first full-featured Java development environment, now unleashes the first Rapid Application Development environment for Java developers:

#### Symantec Visual Café™

Visual Café comes complete with an extensible component library with all of the building blocks you need for your application.

Simply drag and drop a component onto a form. Our Interaction Wizard lets you visually specify all the actions and events. And then Visual Café automatically generates the Java code for you.

Thanks to our exclusive two-way programming you can add or modify the code at the source level, too.

So you'll be whipping out those application prototypes at speeds you can only dream of now.

Imagine building all of your forms visually. Or building your entire user interface without writing one single line of code!

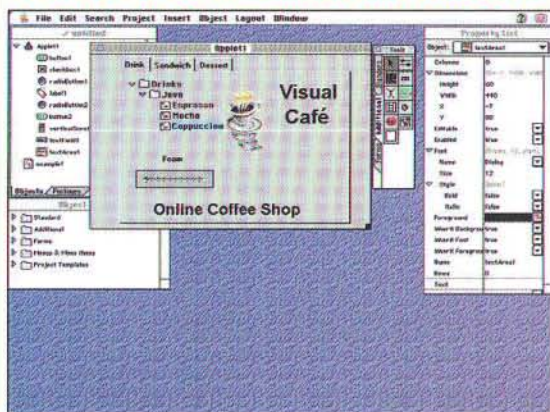
As an added bonus, Symantec's Just In Time (JIT) compiler (included in Netscape's new Navigator™) runs your Java applications faster than any other browser or Java virtual machine on the planet.

So get your hands on the hottest new development tool for Java right away.

For more information, call us at 1-800-453-1059 ext. 9H32 or visit our Web site at [cafe.symantec.com](http://cafe.symantec.com). Or pick up a copy at your local reseller.



### Visual Café for Macintosh® and Windows.™



# SYMANTEC.®



**"Without a doubt, the Premiere Resource Editor for the Mac OS ... A wealth of time-saving tools."**

— MacUser Magazine Eddy Awards

**"A distinct improvement over Apple's ResEdit."**

— MacTech Magazine

**"Every Mac OS developer should own a copy of Resorcerer."**

— Leonard Rosenthol, Aladdin Systems

**"Without Resorcerer, our localization efforts would look like a Tower of Babel. Don't do product without it!"**

— Greg Galanos, CEO and President, Metrowerks

**"Resorcerer's data template system is amazing."**

— Bill Goodman, author of *Smaller Installer* and *Compact Pro*

**"Resorcerer Rocks! Buy it, you will NOT regret it."**

— Joe Zobkiw, author of *A Fragment of Your Imagination*

**"Resorcerer will pay for itself many times over in saved time and effort."**

— MacUser review

**"The template that disassembles PICT's is awesome!"**

— Bill Steinberg, author of *Pyro!* and *PBTools*

**"Resorcerer proved indispensable in its own creation!"**

— Doug McKenna, author of *Resorcerer*



**Until Sept. 1**  
**Only \$128 !**  
**when you order by credit card from our web site.**

# Resorcerer® 2

**Version 2.0**

**The Resource Editor for the Mac™ OS Wizard**

## ORDERING INFO

Requires System 7.0 or greater,  
1.5MB RAM, CD-ROM

Standard price: \$256 (decimal)

**Website price: \$128 - \$256**

(Educational, quantity, or  
other discounts available)

Includes: Electronic documentation  
60-day Money-Back Guarantee  
Domestic standard shipping

Payment: Check, PO's, or Visa/MC  
Taxes: Colorado customers only

Extras (call, fax, or email us):  
COD, FedEx, UPS Blue/Red,  
International Shipping

**MATHEMAESTHETICS, INC.**  
PO Box 298  
Boulder, CO 80306-0298 USA  
Phone: (303) 440-0707  
Fax: (303) 440-0504  
resorcerer@mathemaesthetics.com

**New  
in  
2.0:**

- Very fast, HFS browser generically views any file's contents
- Extensibility for new Resorcerer Apprentices (CFM plug-ins)
- AppleScript Dictionary ('aete') Apprentice Editor
- Integrated support for 'Mcmd' menu commands
- TextTraits Editor
- Complete AIFF sound file disassembly template
- Big-, little-, and even mixed-endian template parsing
- HFS folder attribute editing
- Temporary memory always used everywhere if needed
- Automatically creates 'STR#'s by parsing C/C++ source

- Easier, faster, more Mac-like, and more productive than ResEdit
- Safer memory-based, not disk-file-based, design and operation
- All file information and common commands in one easy-to-use window
- Compares resource files, and even **edits your data forks** as well
- Visible, accumulating, editable scrap
- Searches and opens/marks/selects resources by text content
- Makes global resource ID or type changes easily and safely
- Builds resource files from simple Rez-like scripts
- Most editors DeRez directly to the clipboard
- All graphic editors support screen-copying or partial screen-copying
- Hot-linking Value Converter for editing 32 bits in a dozen formats
- Its own 32-bit List Mgr can open and edit very large data structures
- Templates can pre- and post-process any arbitrary data structure
- Includes nearly 200 templates for common system resources
- TEMPLs for Installer, MacApp, QT, Help, AppleEvent, OCE, GX, etc.
- Full integrated support for editing color dialogs and menus
- Try out balloons, 'icb's, lists and popups, even create C source code
- Integrated single-window Hex/Code Editor, with patching, searching
- Editors for cursors, versions, pictures, bundles, and lots more
- Relied on by thousands of Macintosh developers around the world

To order by credit card, or to get the latest news, bug fixes, updates, and apprentices, visit our website...

**www.mathemaesthetics.com**



# How To Communicate With Us

In this electronic age, the art of communication has become both easier and more complicated. Is it any surprise that we prefer **e-mail**?

If you have any questions, feel free to call us at 805/494-9797 or fax us at 805/494-9798.

If you would like a subscription or need customer service, feel free to contact Developer Depot Customer Service at 800-MACDEV-1

## DEPARTMENTS

**Orders, Circulation, & Customer Service**

**Press Releases**

**Ad Sales**

**Editorial**

**Programmer's Challenge**

**Online Support**

**Accounting**

**Marketing**

**General**

**Web Site (articles, info, URLs and more...)**

## E-Mail/URL

cust\_service@devdepot.com

press\_releases@mactech.com

ad\_sales@mactech.com

editorial@mactech.com

prog\_challenge@mactech.com

online@mactech.com

accounting@mactech.com

marketing@mactech.com

info@mactech.com

http://www.mactech.com

## MacTECH MAGAZINE

*MacTech Magazine is grateful to the following individuals who contribute on a regular basis. We encourage others to share the technology.*

*We are dedicated to the distribution of useful programming information without regard to Apple's developer status.*

*For information on submitting articles, ask us for our **writer's kit** which includes the terms and conditions upon which we publish articles.*

## Editorial Board of Advisors

Scott T. Boyd, Jordan J. Mattson, Jim Straus, and Jon Wiederspan

## Editors

**Publisher** • Neil Ticktin

**Editor-in-Chief** • Eric Gundrum

**Managing Editor** • Jessica Courtney

**Editorial Intern** • Michelle Shin

**Online Support** • Nick "nick.c" DeMello

## Contributing and Technical Editors

**Java** • Will Iverson, Apple Computer, Inc.

**Internet** • Carl de Cordova, Apple Computer, Inc.

**MacOS 8** • Steve Kiene, Mindvision

**MacDev-1™** • Rich Siegel, Bare Bones Software, Inc.

**MagicCap/TeleScript** • Richard Clark

**Components** • Tantek Celik, 6prime Corporation

**Performance Programming** • Jim Gochee, Connectix

**Product Reviews** • Ed Ringel

## Regular Columnists

**Getting Started** • Dave Mark

**Programmer's Challenge** • Bob Boonstra

**From the Factory Floor** • Dave Mark, Metrowerks

**Tips & Tidbits** • Steve Sisak

**MacTech Online/URLs** • Nick "nick.c" DeMello

## XPLAIN CORPORATION

**Chief Executive Officer** • Neil Ticktin

**Chief Operating Officer** • Andrea J. Sniderman

**Controller** • Heather Principe

**Advertising Executive** • Ruth Subrin

**Marketing Manager** • Susan M. Worley

**Customer Relations** • Erik Davidson,

Stephanie Long,

Lee Ann Pham

Susan Pomrantz

**Accounting** • Dennise Gant, Jan Webber

**Financial Services** • Shin Financial Services

**Art Direction/Production** • InfoGraphix

**Board of Advisors** • Steven Geller, Blake Park, and Alan Carsrud



This publication is printed on paper with recycled content.

All contents are Copyright 1984-1997 by Xplain Corporation. All rights reserved. MacTech is a registered trademark and MacDev-1, THINK Reference, Developer Depot, Sprocket, JavaTech, WebTech, BeTech, and the MacTutorMan are trademarks of Xplain Corporation. *develop* is a trademark of Apple Computer, Inc. Other trademarks and copyrights appearing in this printing or software remain the property of their respective holders. Xplain Corporation does not assume any liability for errors or omissions by any of the advertisers, in advertising content, editorial, or other content found herein. Opinions or expressions stated herein are not necessarily those of the publisher and therefore, publisher assumes no liability in regards to said statements.

**MacTech Magazine** (ISSN: 1067-8360 / USPS: 010-227) is published monthly by Xplain Corporation, 850-P Hampshire Road, Westlake Village, CA 91361-2800. Voice: 805/494-9797, FAX: 805/494-9798. Domestic subscription rates are \$47.00 per year. Canadian subscriptions are \$59.00 per year. All other international subscriptions are \$97.00 per year. Domestic source code disk subscriptions are \$77 per year. All international disk subscriptions are \$97.00 a year. Please remit in U.S. funds only. Periodical postage is paid at Thousand Oaks, CA and at additional mailing office.

**POSTMASTER:** Send address changes to **MacTech Magazine**, P.O. Box 5200, Westlake Village, CA 91359-5200.

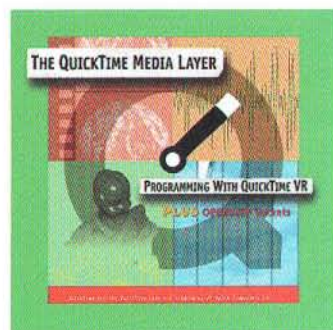


# Contents

July 1997 • Volume 13, No. 7

For Macintosh  
Programmers & Developers

**MacTech**  
M A G A Z I N E



Page 51

## Feature Articles

### MULTIMEDIA

- 43** **Programming With QuickTimeVR**  
A look at the new API for managing QuickTime VR movies  
*by Tim Monroe and Bryce Wolfson*

- 51** **The QuickTime Media Layer**  
An overview of Apple's flagship multimedia software  
*by Tim Monroe*

### TOOLBOX ESSENTIALS

- 12** **Creating Multi-file Dialogs In CPX**  
How to build a custom file dialog with CustomGetFile and how to manage the dialog callback routines  
*by John Shackelford*

### RHAPSODY

- 22** **OPENSTEP Sockets**  
Network programming in the brave new world of BSD Sockets, a networking interface under Rhapsody  
*by Jason Proctor*

- 55** **Rhapsody FAQ's**  
You've heard the rumors, but how much is true?  
*by Michael Rutman*

- 70** **Porting to Rhapsody PPC from OPENSTEP**  
Want your OPENSTEP application to run on more than one platform? Apple is making it easy  
*by Andrew Stone*

### PLUGGING IN

- 35** **Filter It!**  
Join the plug-in revolution: write a FilterTop filter  
*by Alan Weissman*

## Reader Resources

- 72** **NewsBits**  
**74** **Tips & Tidbits**  
**75** **The Classifieds**  
**76** **Advertiser & Product Index**

## Columns

### VIEWPOINT

- 4** **Reality Distorsion Field...Engage**  
*by Eric Gundrum*

### GETTING STARTED

- 6** **Getting To Know NEXTSTEP**  
*by Dave Mark*

### FROM THE FACTORY FLOOR

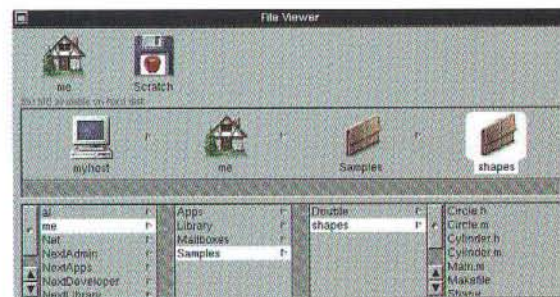
- 31** **The CodeWarrior IDE Team**  
*by Dave Mark*

### PROGRAMMER'S CHALLENGE

- 60** **Disambiguator**  
*by Bob Boonstra*

### URLs

- 68** **Let The QuickTimes Roll**  
*by Nick "nick.c" DeMello*



Getting to Know NEXTSTEP, Page 6

## develop → Special Section

- 77** **Object Support Library Version History**  
*by Andy Bachorski*
- 80** **Newton Q & A**  
*by Apple Developer Support Center*
- 82** **Macintosh Q & A**  
*by Apple Developer Support Center*
- 86** **Mixed Up Threads**  
*by Andy Bachorski and George Warner*



by Eric Gundrum



### REALITY DISTORTION FIELD...ENGAGE

WWDC is over, and I've just spent a full week listening to the official Apple message. Many weeks may have passed by the time you read this article, but my mind is still hashing out the things I heard just a few days ago.

I must admit, I attended WWDC 1997 with pretty low expectations. Last year attendees had Copland forced on them as the only possible future for Macintosh development. This year was very different; Apple listened to developers and offered us several options for our future development. In fact, they even encouraged us to send our comments to <rhapsody-dev-feedback@apple.com>. I am pleased with how well Apple handled the conference.

### Rhapsody Does Not Make Obsolete Programmer's Abilities

Since Apple and NeXT's merger, I have heard many Macintosh developers express concern that their skills programming the Macintosh will be obsolete. After all, Rhapsody replaces the Mac Toolbox with something new, code named Yellow Box. (Yellow Box is the combination of OPENSTEP and key Apple technologies.) The APIs to Yellow Box is dramatically different from the Macintosh Toolbox.

However, what makes a good programmer is not merely knowledge of one system's API. Good programming is about bringing organization to a complex situation, providing a simple solution to solve a complex problem, and identifying what a user wants. Our skills as programmers rely more heavily on our ability to organize the problem and solution than they do on knowledge of any particular API. Anyone can look up API calls in a manual, but it takes an experienced programmer to know what routines to look for and how to connect them together to make a meaningful application.

Some of our greatest skills as Macintosh programmers are our abilities to design meaningful interfaces to help users solve complex problems. These skills can translate to any platform. (One need try only a few Windows applications to see how rare are these skills.) Even though we will need to learn a new programming environment, it also presents a fresh start for new and non-Macintosh programmers to join us. Those who have the high quality skills we have come to expect of Macintosh programmers will do well; the others...well, who knows. Rhapsody offers many new opportunities to build more powerful and interesting tools; I very much look forward to using those tools, and creating some myself.

### MacOS 8 is Alive and Well

Apple had many strong presentations about the future of MacOS 8. They presented a clear schedule for delivering significant upgrades well into 1999. They presented several new technologies being developed for MacOS 8. Most important, they told us Yellow Box would be made to run on MacOS 8 in addition to running on Rhapsody.

Apple is realistic about the adoption rate of Rhapsody; they expect 4-5 million Rhapsody users by mid 1999. That is a far cry from the current 27 million MacOS 7 users. Many developers have complained that they don't want to have to write their applications twice: once to support Rhapsody and a rewrite to support MacOS 8. Apple has responded by porting the Yellow Box APIs to MacOS 8. This will allow developers to use one API to write their code once and have it run on all Macintosh platforms.

This single API approach is especially important given how much we all (developers and Apple) rely on MacOS 8 for our bread and butter now and for the future. Enabling developers to take advantage of Yellow Box to build applications faster than ever before and deploy them on MacOS 8 as well as Rhapsody makes it that much more compelling for us to begin our Rhapsody development as soon as possible.

### Rhapsody Provides Compelling New Opportunities

Many Macintosh developers have been dreaming for some time of entering the Windows software market. We see analysts' numbers that generally have an extra zero or two at the end as compared to our own Macintosh market. (People tend to forget that those extra zeros apply to costs as well as revenues, and to the number of competitors.) I've talked with many developers struggling to deal with porting their software to other platforms. Rhapsody — or more specifically Yellow Box — offers a very different approach. Unlike Java, Yellow Box today offers a much more mature solution to the multiplatform software problem. With so many competing implementations, Java only dreams of solving the problem of write your software once, and deploy it anywhere. With Apple controlling all implementations, Yellow Box is actually delivering it.

Apple has promised Yellow Box compatibility in at least four flavors: Rhapsody for PowerPC, Rhapsody for Intel, Yellow Box for MacOS, Yellow Box for Win32 (that is, NT and 9x). Additionally, there are strong rumors that Yellow Box will also be available for Solaris and HP UNIX. That is a pretty broad market. The goal is that we developers can write our software to one API and merely recompile for each target platform. Apple is even working on multiplatform delivery mechanisms. In reality there are subtle differences between platforms that require some additional tweaks, but Apple is helping even there. For example, there is a specific nib file (resource fork) for each target platform, allowing platform-specific interface changes to override the platform's default behavior.

Another reason to develop for Yellow Box is the completeness of the application framework. We have had frameworks for Macintosh for many years now, but none fully hide the need to know the underlying Mac Toolbox. Yellow Box is the toolbox. When building an Yellow Box application, so

*continued on page 59*





# MORE DEVELOPERS PROTECT.



## MachASP Packs More Into Less.

Based on state-of-the-art ASIC technology, MachASP packs the industry's most advanced security, compatibility and flexibility into a compact and end-user-friendly dongle.

## Grow With Aladdin!

The fastest growing company in the industry, with over 4 million keys sold to 20 thousand developers worldwide, Aladdin is setting the standard for software security today.



## NSTL Study Rates HASP No.1!



A recent test conducted by the National Software Testing Labs<sup>1</sup>, the world's foremost independent lab, compared the flagship PC products of leading software protection vendors. The result? HASP was rated the clear overall winner - and number one in all the major comparison categories. For a full copy of the NSTL report, contact your local HASP distributor.

# MachASP<sup>®</sup> PROTECTS MORE.



Mac OS

These days, more and more developers are choosing to protect their software against piracy. They're protecting more products, on more platforms, with better protection – and selling more as a result.

And more of these developers are protecting with MachASP. Why? Because MachASP offers more security, more reliability and more features than any other product on the market. Only MachASP offers capabilities such as network support, anti-debugging envelope protection, and secure remote activation and updating.

MachASP supports the most advanced platforms, including all versions of MacOS and Power Mac – as well as AppleTalk networks. And because Aladdin is a licensed Apple Partner, MachASP guarantees full, transparent compatibility with the ADB standard.

To learn more about how you can protect better – and sell more – call now to order your MachASP Developer's Kit.



*The MachASP Developer's Kit contains everything you need to protect your software today!*

**1-800-223-4277**  
www.aks.com

**ALADDIN<sup>™</sup>**

*The Professional's Choice*

**North America** Aladdin Knowledge Systems Inc. Tel: (800) 223-4277, 212-564-5678, Fax: 212-564-3377, E-mail: hasp.sales@us.aks.com  
**Int'l Office** Aladdin Knowledge Systems Ltd. Tel: +972-3-636 2222, Fax: +972-3-537 5796, E-mail: hasp.sales@aks.com  
**Germany** FAST Software Security AG Tel: +49 89 89 42 21-37, Fax: +49 89 89 42 21-40, E-mail: info@fast-ag.de  
**United Kingdom** Aladdin Knowledge Systems UK Ltd. Tel: +44 1753-622266, Fax: +44 1753-622262, E-mail: sales@alldn.co.uk  
**Japan** Aladdin Japan Co., Ltd. Tel: +81 426-60 7191, Fax: +81 426-60 7194, E-mail: sales@aladdin.co.jp  
**Benelux** Aladdin Software Security Benelux B.V. Tel: +31 24-641 9777, Fax: +31 24-645 1981, E-mail: 100526.1356@compuserve.com

■ Aladdin Russia 095 970558 ■ Australia Conlab 03 96905605 ■ Chile Micrologica 02 7350041 ■ China Shanghai LBB 021 64377628 ■ Czech Atlas 02 766085 ■ Denmark Berendsen 029 577316 ■ Egypt Zokeldin 02 3604632 ■ Finland ID Systems 0 8703520 ■ France 1 40858865 ■ Greece Unibrain 01 6756320 ■ Hong Kong Hartings 02 5484629 ■ India Solution 011 2148254 ■ Italy Partner Data 02 76147380 ■ Korea Dae-A 02 8484401 ■ Mexico SIGSA 91800 55783 ■ New Zealand Training 04 5666014 ■ Poland Synthem 061 480273 ■ Portugal Iulianica 01 4116269 ■ Romania Bio Interactive 064 140203 ■ Singapore ITI 065 5666788 ■ South Africa D Le Roux 011 8864704 ■ Spain PC Hardware 03 4493193 ■ Switzerland Opag 061 7169222 ■ Taiwan Teco 02 5550676 ■ Turkey Mikrolisa 0312 4670635 ■ Yugoslavia Apps 021 623920

© Aladdin Knowledge Systems Ltd. 1995-1996. (10) 980 MacASP<sup>™</sup> is a registered trademark of Aladdin Knowledge Systems Ltd. All other product names are trademarks of their respective owners. Mac & the Mac OS logo are trademarks of Apple Computer, Inc., used under license. NSTL makes no recommendation or endorsement of any product. The NSTL report was commissioned by Aladdin.





by Dave Mark, ©1997, All Rights Reserved

# Getting to Know NEXTSTEP

Do you remember the very first time you sat down at a Macintosh? For me, it was almost a religious experience. I instantly connected to the interface, to the elegance of the Mac look-and-feel. Recently, I've had the opportunity to explore some of the other major operating systems out there, including, most recently, NEXTSTEP. I must say I was both surprised and impressed.

Why spend time with NEXTSTEP? As you know, Apple bought NeXT with the intention of basing their next OS, Rhapsody, on the NEXTSTEP technology. By the time you read this, Apple should be putting the finishing touches on a preliminary release of Rhapsody. Till then, NEXTSTEP is the closest thing I have to Rhapsody. Also, I think it is worthwhile exploring the roots of our new OS, to get a sense of the logic behind some of the Rhapsody design decisions.

NEXTSTEP is both nice looking and easy to use. Though I did have trouble installing the developer's release on my PC, once it was up and running I found NEXTSTEP to be solid and a joy to use. This month, we'll take a first look at NEXTSTEP and the process of building a console Objective-C app. We'll also look at Objective-C's category mechanism, a cool way of seamlessly extending an existing class.

This month's column owes a debt of gratitude to CodeWarrior Gene Backlin. Gene wrote the book *Developing NEXTSTEP Applications* (ISBN# 06723-06581) and helped me come up to speed on the NEXTSTEP interface and the Project Builder and Interface Builder applications. Thanks, Gene!

And thanks also to Michael Rutman for his monthly expert tech review. As always, the mistakes are all mine...

## WORKING WITH NEXTSTEP

One of the first things you'll notice when you boot up under NEXTSTEP is the "dock" that appears in the upper-right corner of your screen. My dock is shown in **Figure 1**. Each square in the dock represents an NEXTSTEP application. The NeXT cube at the top of the dock represents the NEXTSTEP equivalent of the Finder, called the FileViewer. Below that is the preferences application (which I've set to show the time and date). The last three squares are a terminal emulator which you can use to run your console apps, and the two programs which make up the NEXTSTEP development environment, Project Builder (the hammer) and Interface Builder (the screwdriver). We'll use the terminal emulator to build this month's apps, and move on to Project Builder and Interface Builder next month.



**Figure 1.** The NEXTSTEP "dock". By default, it appears in the upper right corner of the NEXTSTEP screen.

To launch an app from the dock, double-click on its square. If the app is already running, a double-click will bring it to the foreground, just like a Mac. You can tell if an app is not running by the elipsis (...) in the lower right corner of its dock square (i.e., no ellipsis means the app is running).



wide  
sesame  
dec  
mind  
season

architecture

for business

possibilities

horizons

house

now

platform

window

road

sky

borders

up

# Open text without boundaries

WITHOUT US THEY  
WOULDN'T EXIST  
(OR AT LEAST THEY'D  
BE MUCH MORE  
FRUSTRATED.)

America Online

Apple Computer

Broderbund Software

Canon

Claris

CompuServe

Dataware

Digital Equipment

Fujitsu

GE Information Systems

ichat

Macromedia

National Geographic

Now Software

QualComm

Seagate Software

3M

US West

Xerox

**CONSIDER** the possibilities. They're wide open. Simply put, OpenPaige™ can do anything you want to the text of your software application.

Across platforms, across architectures, even UNIX™. OpenPaige is a feature-rich text environment. Every aspect of text and layout formatting, editing and display is possible. Stylized text. Scaling. Container and non-rectangular shapes. Style sheet support. Cut, copy and paste. Embedded pictures, objects, buttons and even calls to other applications are at your fingertips.

Don't waste time struggling with your text. We've already done the work for you. Drop in OpenPaige and go. Use as much or

as little of it as you wish. Available as object code or as source code, it has an elegantly small footprint. Yet it will empower you to open doors, boundaries, and minds with your software application.

But don't take our word for it. Check out our extensive and diverse client list. They've all discovered how the robust and flexible power of OpenPaige can open up entire new worlds of possibilities.

Open  
Paige™

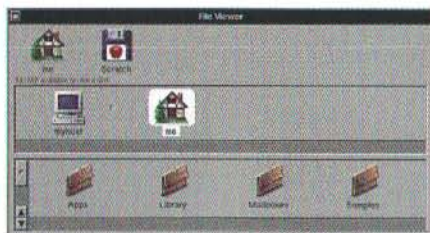
Datapak

For additional information, check out our web site at [www.datapak.com/~datapak/](http://www.datapak.com/~datapak/) or call us at 1-800-327-6703.

Copyright ©1997 Datapak Software. Paige and OpenPaige are trademarks of Datapak Software. All brand or product names are trademarks or registered trademarks of their respective holders. Datapak Software 11815 NE 99th Street, Suite 1200, Vancouver, WA 98682 • Phone: 360.891.0542 • Fax: 360.891.0743 • E-mail: [sales@datapak.com](mailto:sales@datapak.com)



In general, you'll use the File Viewer to wander around your hard drive. As you can see in **Figure 2**, the File Viewer window is divided into three areas. The top area is for mounted volumes and aliases. The central strip shows where you are currently. In this case, we are looking at the directory /myhost/me. The bottom area shows the files and folders in the current directory. In this case, the directory /myhost/me contains 4 folders and 0 files.



**Figure 2.** The File Viewer window, showing the files in the default, "me" directory.

Notice the apple icon in the floppy disk in the top line? Yup. I stuck a Mac floppy in the PC floppy drive and NEXTSTEP recognized it. Very cool! That's how I was able to get my source code and screen shots (shot with the Grab application) over to my Mac. There's something nice about being able to work with a Mac floppy on my PC...

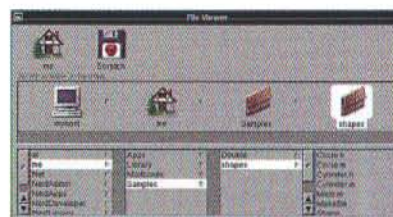
**Figure 3** shows a pair of menu windows. The left window (labeled Workspace) holds the main File Viewer menu. The right window (labeled View) appeared when I clicked on the View item in the Workspace menu. Menus and submenus. Like a Mac, but different. Menu items can have command-key equivalents, but since this version of NEXTSTEP was running on my PC, they were Alt key equivalents. In general, NEXTSTEP feels like it borrowed heavily from the Mac (and why not?). If you are not sure how to do something, think about how you'd do it on a Mac. For example, Alt-c and Alt-v copy and paste, Alt-q quits, and Alt-w closes the frontmost window. Sound familiar?

Workspace	View
Info	Browser
File	Icon
Edit	Listing
Disk	Sort Icons
View	Clean Up Icons
Tools	New Viewer
Windows	Update Viewers
Services	
Hide	
Log Out	

**Figure 3.** The Workspace menu and View submenu.

NEXTSTEP allows you to remap the keyboard. By default, on an Intel box, the left Alt-Key is mapped to be a command key. There are several different configurations built in, and you can create your own.

**Figure 4** shows the File Viewer with the Browser View selected. The NeXT browser is a truly nice way to navigate your file system. I hope this part of the NEXTSTEP interface makes it into Rhapsody. Note that as you descend into a directory, the path is reflected in the middle strip, while the details at each level are displayed in the browser portion at the bottom of the File Viewer window. In **Figure 4**, we are looking at the shapes directory, which holds the files from last month's sample program.



**Figure 4.** The File Viewer, with the Browser view.

### RUNNING LAST MONTH'S EXAMPLE

As I brought my source code over from my Mac (where I was running CodeBuilder), I learned a few important lessons. First, try to avoid spaces in your file names. If you include a space in a file or directory name, you'll create a lot of extra work (and potential hair pulling) for yourself as you add escaped quotes to your makefiles, etc. to make sure the spaces are kept in the names. Save yourself the trouble and leave the spaces out.

I'm not sure of the details, but I've found that some odd (invisible) characters creep into my code when porting from Mac to NEXTSTEP. This could be a result of carriage return/line feeds being treated differently on the two platforms, or an artifact caused by my method of porting the files across. Regardless, if you find a line of code that looks perfect, yet behaves oddly, make sure that your files are true Unix files. If you create them in CodeWarrior or BBEdit, or on the NeXT box, this shouldn't be a problem. If you created your file on another platform (like your Mac), the mechanism you used to move the file to your NeXT box should handle the translation for you.

If all else fails, you might try translating your source file using the command:

```
tr '\015' '\012' < sourceFile > newFile
```

Another lesson learned is that different versions of OPENSTEP and NEXTSTEP look and behave quite differently from each other and from Tenon's CodeBuilder environment. For example, CodeBuilder doesn't support #import (at least not in the same way as NEXTSTEP). Under CodeBuilder, I had to use a #ifndef to avoid recursive inclusion of my header files. Moving to a genuine NEXTSTEP environment was really nice.



# Behind every good Mac app...

Every job requires the proper tool. When it comes to crafting good code, **BEdit 4.0** is the **Swiss Army Chainsaw** of text editors.

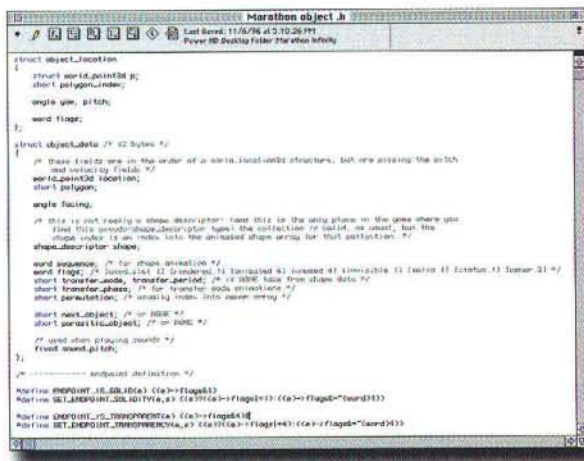
**"If you have to spend time with text on the Mac, you should use BEdit."**

— MacWeek, September 30, 1996

- Compare source files and apply differences
- Open From and Save To FTP
- Search and replace multiple files using Grep
- Navigate source files with integrated **PopupFuncs™** technology
- Browse project documents



Bungie Software's **Marathon Infinity**, created using **BEdit**.



Source code from **Marathon Infinity** displayed in **BEdit**.

...is a **great**  
text editor.

## BEdit 4.0



<http://www.barebones.com/chainsaw>

To order, call 617-778-3100



**Bare Bones Software, Inc.**

P.O. Box 1048, Bedford, MA 01730 • main 617-778-3100 • fax 617-778-3111



# SmalltalkAgents®

Integrated Development Environment

for

Macintosh  
Windows 95/NT



Check out what's new...

[www.SmalltalkAgents.com](http://www.SmalltalkAgents.com)



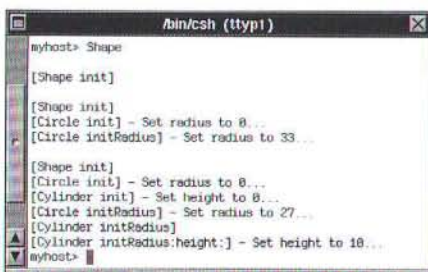
1405 Main Street  
P.O. Box 371478  
Montara, CA 94037-1478 USA

© Quasar Knowledge Systems, Inc. 1997  
Windows 95 and Windows NT are Trademarks of Microsoft.

There are other differences as well. CodeBuilder calls gcc, while under NEXTSTEP, gcc is called cc. This requires a small change in the makefiles. Also, NEXTSTEP automatically includes the Objective-C library in its standard libraries, so there is one less library to include. Here's the makefile I used when I moved Shape over to NEXTSTEP:

```
Shape: Main.m Shape.m Circle.m Cylinder.m
cc -o Shape Main.m Shape.m Circle.m Cylinder.m
```

Note the use of cc instead of gcc as well as the absence of the link library. **Figure 5** shows the result, running in a terminal emulator window. Once we move into Project Builder and Interface Builder (next month's column) you can say goodbye to all this console business.



**Figure 5.** Running last month's Shape application inside the terminal emulator.

## ADDING CATEGORIES TO YOUR CODE

Objective-C features a mechanism known as a category that allows you to extend an existing class without modifying the .m and .h file of the original class. Why would you want to do that? The most obvious reason would be to extend a class when you don't have access to the source that defines the class. But there are other reasons to use a category to extend a class. You might want to organize your class into related subgroups. This is especially useful if you are building an extremely large class (you can take advantage of incremental compilation) or if your class is being worked on by more than one person. You might also use categories to extend a class in different ways for different applications.

To implement a category, you'll define a set of methods placing the interface in a header file and the implementation in a .m file. You can use a category to add methods to a class, but you are *not* allowed to use a category to add any variables to the class. A category method *can* override an inherited method and can get at the inherited method by sending a message to super. But a category method has no way to call a method in the original class having the same name as the category method (i.e., don't create a category method with the same name as a method in the class being extended).

An example should help make this clear. If you remember, last month's example implemented 3 classes: Shape, Circle, and Cylinder. Shape is derived from Object, Circle from Shape, and Cylinder from Circle. Without modifying the Shape, Circle, or Cylinder classes, we'll extend the Circle class and access the new methods from a Cylinder object. We'll call the category CircleMath. It will add an area and circumference method to the Circle class.

Here's the source code for CircleMath.h:

```
#import "Circle.h"

@interface Circle (CircleMath)
- (float)area;
- (float)circumference;
@end
```

Here's the source code for CircleMath.m:

```
#import "CircleMath.h"

@implementation Circle (CircleMath)

- (float)area
{
    return ((float)radius * (float)radius * 3.14159);
}

- (float)circumference
{
    return ((float)radius * 2.0 * 3.14159);
}

@end
```



Cross-Platform Object Database Engine

# neoAccess

ORDER NOW!  
version 5.0 is here!  
**5**

**Order directly from web site at [www.neologic.com](http://www.neologic.com) and save \$100!**

Download the architectural overview or view it on-line with Adobe Acrobat 3.0

## The most powerful object-oriented database engine available

NeoAccess® displays electrifying performance — up to ten times that of the competition. Behind its elegant programming interface is a fully optimized relational query engine built for speed. It also has an object cache with automatic garbage collection so your applications can run in a much smaller memory footprint than you ever imagined possible. And NeoAccess has no runtime fees so you pay one affordable price no matter how many copies of your application you use or sell.

While others promise cross-platform — NeoAccess delivers. NeoAccess is a set of C++ classes designed for use with popular compilers and frameworks on Windows®, Macintosh®, and Unix® platforms. Thousands of developers, including America Online® and Claris®, have already found that NeoAccess enabled them to build fast, powerful internet applications in record time. That's why there are more NeoAccess based applications on end-user machines than any other object database backend.

M-F 9AM to 5PM Pacific:  
**1-800-919-6353**

For Information and Customer Service:  
**1-510-524-5897**

**neo•logic®**

NeoLogic Systems, Inc.  
1450 Fourth St., Suite 12, Berkeley, CA 94710  
V. 510.524.5897 F. 510.524.4501  
[neologic@neologic.com](mailto:neologic@neologic.com)

Here's the new version of Main.m. We've added a #include of CircleMath.h as well as a pair of printfOs. The first printfO sends an area message to cylinder and the second printfO sends a circumference message to cylinder.

```
#include "Cylinder.h"
#include "CircleMath.h"

void main()
{
    id shape = [[Shape alloc] init];
    id circle = [[Circle alloc] initWithRadius:33];
    id cylinder = [[Cylinder alloc] initWithRadius:27
height:10];

    printf( "\ncylinder area = %f...\n",
[cylinder area] );
    printf( "[cylinder circumference] = %f...\n",
[cylinder circumference] );

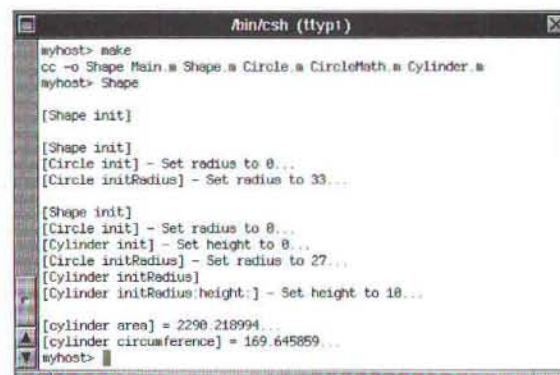
    [shape free];
    [circle free];
    [cylinder free];
}
```

Finally, here's the make file I used:

```
Shape: Main.m Shape.m Circle.m CircleMath.m Cylinder.m
cc -o Shape Main.m Shape.m Circle.m CircleMath.m Cylinder.m
```

Note that I added the file CircleMath.m to both lines. That's it.

**Figure 6** shows the output when I ran this new version of Shape.



```
Min/csh (tty1)
ayhost> make
cc -o Shape Main.m Shape.m Circle.m CircleMath.m Cylinder.m
ayhost> Shape

[Shape init]

[Shape init]
[Circle init] - Set radius to 0...
[Circle initWithRadius] - Set radius to 33...

[Shape init]
[Circle init] - Set radius to 0...
[Cylinder init] - Set height to 0...
[Circle initWithRadius] - Set radius to 27...
[Cylinder initWithRadius]
[Cylinder initWithRadius,height:] - Set height to 10...

[cylinder area] = 2290.218994...
[cylinder circumference] = 169.645859...
ayhost>
```

**Figure 6.** Running Shape with the CircleMath category added in.

## TILL NEXT MONTH...

I am very jazzed about NEXTSTEP and the promise it holds for Rhapsody. I can't wait to get to WWDC (a few weeks from now) and find out about what portions of the NEXTSTEP framework will make their way into Rhapsody. Till then, we'll continue to expore NEXTSTEP and, perhaps, make a return to Java as well. Got an opinion? Drop me a line... **MT**





# Creating Multi-File Dialogs in CPX

## *How to build a custom file dialog with CustomGetFile and how to manage the dialog callback routines*

### INTRODUCTION

The ease that applications are built in CPX often masks the flexibility of writing low-level code for the Macintosh. Prograph CPX ships with a class library called Application Building Classes (ABCs) which, in the world of Macintosh application frameworks, ranks among the richest. However, one particularly useful chunk of code that you will not find in the ABCs is the fancy file dialog you create using CustomGetFile.

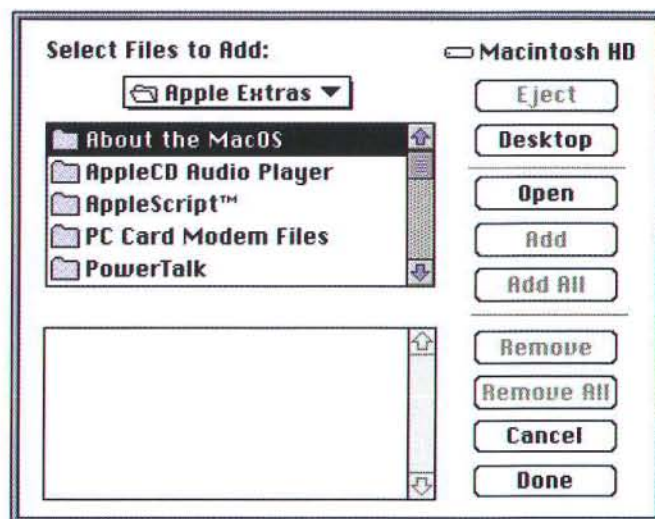
You use the CustomGetFile feature to add source files to projects in several popular Macintosh applications like CodeWarrior, Roaster or Symantec Project Manager. The File classes in the ABCs all use the Standard GetFile dialog to retrieve files. This is fine for single file operations, but if your application must provide the ability to manipulate several files at once, CustomGetFile must be used to add controls for collecting the list of files. Surprisingly, when I wrote the code for a CPX version of CustomGetFile, it was difficult to find a complete example in C or Pascal on which to base the design. I

did find some chunks or pieces in C and some CPX that I used as a basis for MultiFile.

MultiFile is a CPX section. It provides a class for a multifile dialog of the type shown below. It allows you to navigate, add and subtract several files all in one dialog session. The section also provides a basic file dialog allowing you to specify the user prompt. However, this article focuses on MultiFile Dialog — it shows you how to use the MultiFile Dialog class and discusses key code snippets.

### MULTIFILE DEMO APPLICATION

The demo application opens a MultiFile Dialog and allows you to add and subtract files. Its purpose is to demonstrate what a MultiFile Dialog provides (see **Figure 1**). The demo application is one of those *minimal* apps. It does not use the ABCs and has no value other than showing you how the MultiFile Dialog looks.



**Figure 1.** The Demo Application opens a MultiFile Dialog.

**John Shackelford** is the founder of Tangent Systems — a software development company based in San Diego. When he's not playing with his 3 children, he's busy writing CPX code or playing the piano. He can be reached at <shackx@aol.com>. You can find Tangent Systems on the world wide web at <<http://www.tangentsys.com>>.



The Macintosh Toolbox provides a function called CustomGetFile for creating custom file dialogs. First, I'll show you the top level CPX code for using it. That way, if all you want is to use MultiFile and you don't care about how it works, you'll have all you need to know. I will also show the function prototype and build up the associated functions for it in CPX.

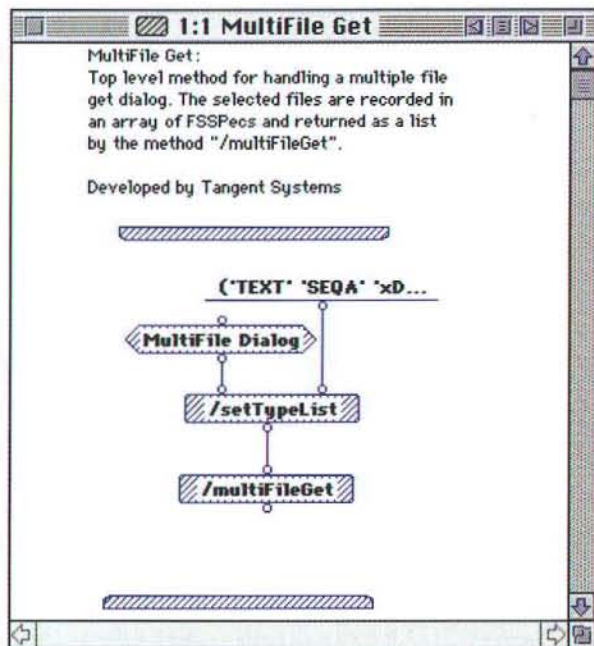


Figure 2. The MultiFile section comes with an example method for using MultiFile.

Using MultiFile is pretty straightforward — explaining it will be much more difficult. However, as you can see in **Figure 2**, all you do is create an instance of MultiFile Dialog, setup the type of files you want with the message `"/setTypeList"` and finally send the message `"/multiFileGet"` to get the whole process rolling. When the method completes, the list of selected files appears on the output of `"/multiFileGet"`.

### HOW IT ALL WORKS

I'll first decompose the main method `"/multiFileGet"`, describe the attributes of MultiFile Dialog and then expose in detail the major chunks of code that makes MultiFile Dialog work in support of CustomGetFile — the Toolbox call we are encapsulating.

#### MultiFile Dialog Constructor

The first thing to look at is the constructor for MultiFile Dialog (**Figure 3**). When an instance of MultiFile Dialog is created, several attributes are initialized. The "Files" attribute will end up holding a new instance of File Results. This object will eventually hold the FSSpecs for the selected files, and it provides methods to add and remove FSSpecs from its internal list of FSSpecs. The attribute "Select List" is set to NULL. Eventually, it will hold a pointer to the scroll list item created by a call to LNew. The Persistent called "The

Dialog" is set to the current MultiFile Dialog instance. Two other methods are executed — one creates a Standard File Reply record while the other (`"/makeCallbacks"`) sets up pointers to the callback functions. So by design a "The Dialog" (global) holds the MultiFile Dialog instance. Everything else hangs off of that — the lower scroll list, StandardReplyRec and the File Results object. The callbacks rely on "The Dialog" to access the scroll list, the File Results object and the StandardReplyRec.

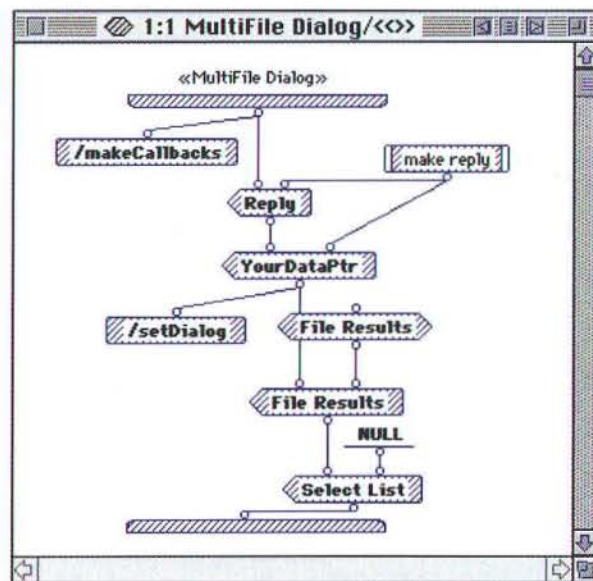


Figure 3. The "Constructor" for MultiFile Dialog sets up 3 Persistants.

The method `"/makeCallbacks"` (**Figure 4**) checks each of 4 attributes in MultiFile Dialog and creates a pointer to a callback method if a method name is specified. This provides a nice way for users of the class to specify their own callback methods — just by specifying their own methods in the initialization list that can be fed to the Constructor. And of course if that does not provide enough flexibility, you can always subclass the whole thing and define `"/makeCallbacks"` anew.

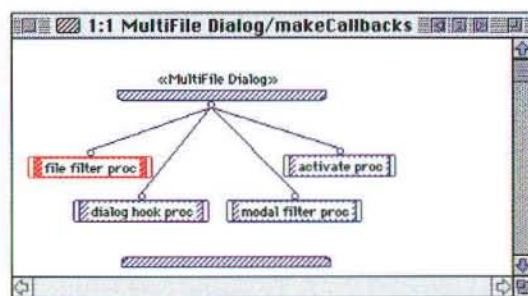
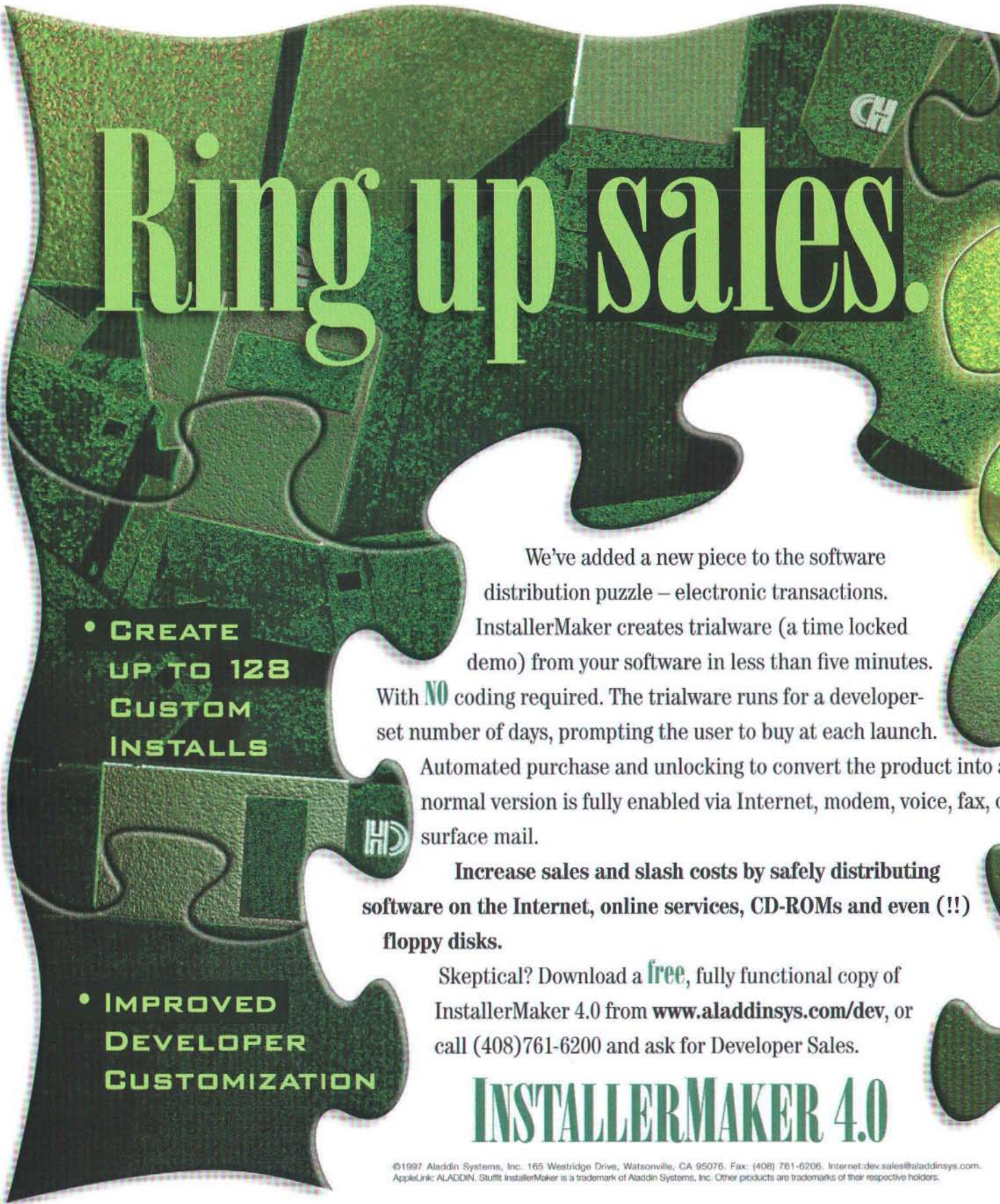


Figure 4. `"/makeCallbacks"` checks each of 4 MultiFile Dialog attributes.

continued on page 16





# Ring up sales.

- **CREATE  
UP TO 128  
CUSTOM  
INSTALLS**

- **IMPROVED  
DEVELOPER  
CUSTOMIZATION**

We've added a new piece to the software distribution puzzle – electronic transactions.

InstallerMaker creates trialware (a time locked demo) from your software in less than five minutes.

With **NO** coding required. The trialware runs for a developer-set number of days, prompting the user to buy at each launch.

Automated purchase and unlocking to convert the product into a normal version is fully enabled via Internet, modem, voice, fax, or surface mail.

**Increase sales and slash costs by safely distributing software on the Internet, online services, CD-ROMs and even (!!)** floppy disks.

Skeptical? Download a **free**, fully functional copy of InstallerMaker 4.0 from [www.aladdinsys.com/dev](http://www.aladdinsys.com/dev), or call (408)761-6200 and ask for Developer Sales.

## INSTALLERMAKER 4.0

©1997 Aladdin Systems, Inc. 165 Westridge Drive, Watsonville, CA 95076. Fax: (408) 761-6206. Internet: dev.sales@aladdinsys.com. AppleLink: ALADDIN. Stufft InstallerMaker is a trademark of Aladdin Systems, Inc. Other products are trademarks of their respective holders.





- **INSTALL OR  
UNINSTALL**

- **BUILT-IN  
RESOURCE  
COMPRESSION**

- **CUSTOM  
DESTINATIONS**

- **BUILT-IN  
UPDATERS**

- **FULL SCRIPTING  
AND RECORDING**

- **RESOURCE  
INSTALLATION**

- **MOVE/COPY/RENAME  
ANY FILE**

 **Aladdin  
Systems**





The local method "file filter proc" (Figure 5) is typical of the calls in "/makeCallbacks". An improvement on this local method would be to verify that the universal method is actually defined and if it isn't, then fail into a next case and set the "callback" attribute (in this specific case "File Filter Callback") value to NULL.

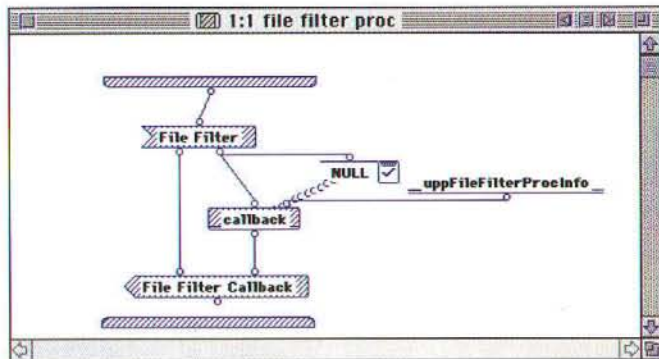


Figure 5. This local method is typical for what a "/makeCallbacks" method does.

The method "/multiFileGet" (Figure 6) is composed of three methods itself. The first performs the Toolbox call, the second retrieves the selected files and the final method cleans up.

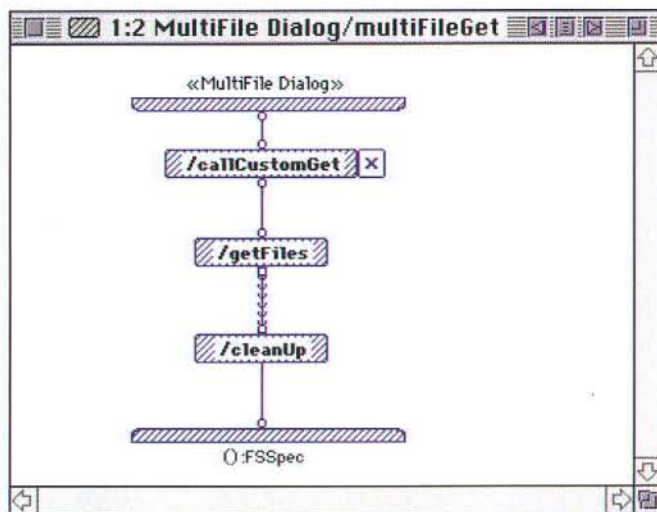


Figure 6. "/multiFileGet" retrieves a list of file specifications.

#### /callCustomGet

The method "/callCustomGet" (Figure 7) makes the Toolbox call "CustomGetFile". The MultiFile Dialog instance carries the values necessary for the function to operate as the user wants.

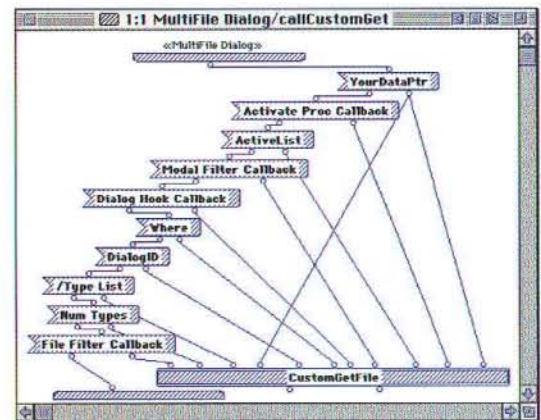


Figure 7. "/callCustomGet" makes the low level Toolbox call.

Let's first look at the CustomGetFile arguments and what Prograph uses. The calling arguments for CustomGetFile are shown below (**bold**) with the MultiFile Dialog Attribute (underlined):

#### **FileFilterYDProcPtr** FileFilter Callback

A pointer to an optional file filter function, provided by your application, through which CustomGetFile passes files of the specified types.

#### **short** Num Types

The number of file types to be displayed. If you specify a "Num Types" value of -1, the first filtering passes files of all types.

#### **SFTypeList** /Type List

A list of file types to be displayed. You can define it using the call "/setTypeList" which automatically sets the value of attribute "Num Types" used above.

#### **StandardFileReply** \* YourDataPointer

The reply record, which CustomGetFile fills in before returning. We create a pointer to a Reply record in the Constructor for MultiFile Dialog.

#### **short** DialogID

The resource ID of a customized dialog template. To use the standard template, set this parameter to 0. The MultiFile Dialog defines this value to be 1001. If you want to customize the look of the dialog, create a new dialog resource and set the DialogID to the new value in a subclass of MultiFile Dialog for the "new" dialog.

#### **Point** Where

The upper-left corner of the dialog box in global coordinates. This value is predefined to be [-1, -1] which will place it in the middle of the screen.

#### **DlgHookYDProcPtr** DialogHook Callback

A pointer to the dialog hook function, which handles item selections received from the Dialog Manager. This is in many ways the "meat" (I suppose I should instead say "fiber" for those vegetarians among us) of the matter; clicks within the dialog get handled by this method.

#### **ModalFilterYDProcPtr** ModalFilter Callback

A pointer to your modal-dialog filter function, which determines how the Dialog Manager filters events when called



You already know that...

**MicroGuard** Copy Protection is -

# UNBEATABLE



**So... Here's how you can reach us:**

**[www.micromacro.com](http://www.micromacro.com)**

International

Micro Macro Technologies, Ltd.  
3 Hashikma St.  
P.O.Box 11516, Azur 58001  
Israel

Tel: (972-3) 558-2345  
Fax: (972-3) 558-2344  
E-mail: [info@micromacro.com](mailto:info@micromacro.com)

USA

MicroGuard  
631 South Pontiac St.  
Denver CO. 80224  
USA

Tel: (303) 320-1628  
Fax: (303) 320-1599  
E-mail: [usa@micromacro.com](mailto:usa@micromacro.com)



# MICROGUARD **PLUS.**





by CustomGetFile. Specify a value of NIL if you are not supplying your own function. We need a modal-dialog filter function because we must provide scroll control for our scrolling list. This method handles clicks in the user-defined lower scroll list. (Control of the upper list “comes” with CustomGetFile.)

**short** ActiveList

**ActivateYDProcPtr** Activate Proc Callback

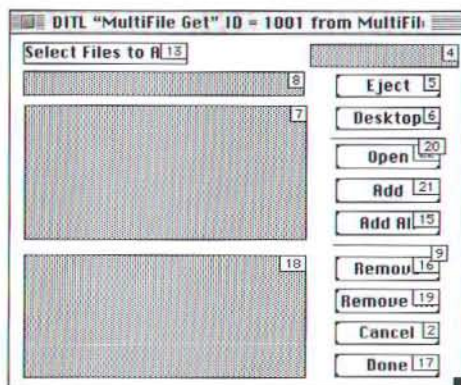
**void \*** YourDataPointer

These last three parameters are not used, and are set to NULL.



**Figure 8.** The default attributes are defined to work with a dialog resource 1001.

**Figure 9** shows the default MultiFile dialog (ID = 1001) and the index numbers for its items.

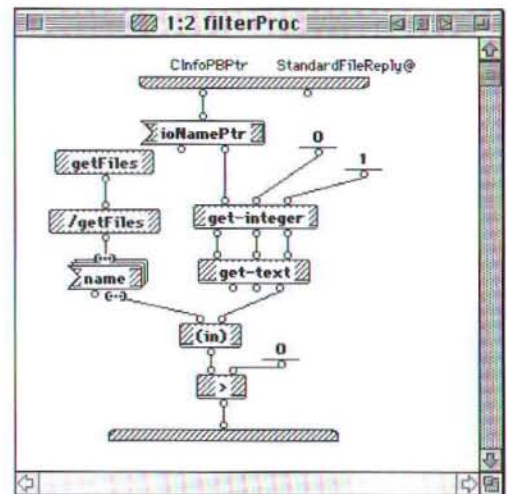


**Figure 9.** Default MultiFile dialog (ID = 1001).

When CustomGetFile is called, three methods (all are Universals) take over until the user clicks “Cancel” or “Done” in the dialog window. They are “filterProc”, “modalFilterYD” and “dialogHook”. We’ll describe each in turn.

## FILTER PROC METHOD

The “filterProc” method (**Figure 10**) does one job. It is called whenever the upper scroll list is about to be filled with files (after passing through the internal “types” filter). It gives us a chance to do further processing. The output of this method is a Boolean which determines whether or not the file should appear in the upper list. A TRUE output removes the item from the upper list while a FALSE value allows it to appear in the upper list. This method checks the list of FSSpecs in the File Results object. If the file is in the File Results object, this filter will remove it from the upper list.



**Figure 10.** “filterProc” is called whenever the upper scroll list is about to be filled with files.

The field “ioNamePtr” holds a pointer to a Pascal string. The first byte in a Pascal string must be the length of the string. we execute “get-integer” to determine the length of the string, and then we execute “get-text” using that value and start at byte 1 of the buffer. We then check the list of file names held by the File Results object (retrieved with “getFiles”). If a matching file name is found, the primitive “(in)” will return an integer greater than 0 indicating the files position in the list. In that case the method outputs TRUE. Otherwise it will output FALSE.

**Want to suggest an article for the magazine? Send your suggestion to**  
**<mailto:editorial@mactech.com>**



Purify on Unix  
Bounds Checker on Windows  
Now, from the makers of QC, comes...



# SPOTLIGHT<sup>TM</sup>

On Macintosh

## Find Bugs Fast

Spotlight is the first **Automatic Memory Debugger** for the Macintosh. Instantly detect invalid memory accesses, bad toolbox parameters, leaks, stack overwrites, memory relocation problems, and much more.

Spotlight uses your XSYM file to automatically patch your application — no need to change your source code. No need to recompile. **No learning curve whatsoever.**

The interface gives you instant feedback when an error is detected. You can ignore the error, ignore all future occurrences of the error, or log the error to a text file for later analysis.

## Fine Grained Memory Protection

Spotlight identifies reads and writes outside of the application and system heap. But it does not stop there. Spotlight's Object Code Replacement instrumentation **inspects each read and write instruction in your code**, detecting faulty memory reads and writes that occur between blocks, in released blocks, across multiple blocks, and in ROM.

This technology works on any heap object you can allocate: Mac heap objects, C 'malloc' heap objects, even C++ objects created with new.

Spotlight stops your application whenever a faulty memory access is about to occur and displays the **exact source code line** where it will happen. The calling stack is shown along with a display of all variables. A memory viewer shows the erroneous memory address and contents.

## Toolbox Validation

Spotlight checks parameters to over **400 toolbox API calls**, automatically validating handles, memory blocks, return values, and so on. Specific checks catch subtle errors such as drawing into an unlocked GWorld, passing an invalid window

pointer, passing an address within an unlocked handle to a routine that may move memory, and too many more to list.

## Leak Detection

No more struggling with MacsBug. On program exit Spotlight provides a **clear listing of all leaked memory** showing a full stack trace from where the memory was allocated. Instantly discover all leaked Mac OS objects, malloc'd objects, C++ objects, and resources.

## Limitations

Spotlight requires an XSYM file to function. MPW and CodeWarrior users can generate these directly. Symantec users must use ToolServer to link with an XSYM capable linker. Spotlight requires a PowerPC processor.

Spotlight works on the object code. **No source code is required:** just the thing for testing purchased third party libraries.

## Availability and Pricing

Pricing for Spotlight DR1 is \$199 US (plus \$5 shipping and handling within the continental US, \$15 for international orders). This includes a free upgrade to the GM version and access to ftp interim upgrades leading up to GM. QC users can cross-grade to Spotlight for only \$149.

All Onyx products carry a 30 day no questions asked money back guarantee.



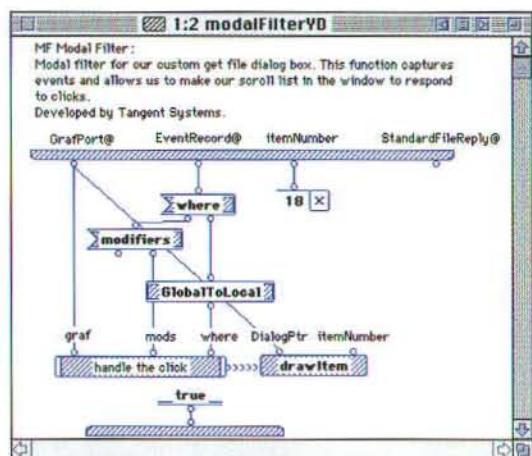
**Onyx Technology, Inc.**  
7811 27th Avenue West  
Bradenton, Florida 34209

sales@onyx-tech.com    www.onyx-tech.com  
941 795-7801    941 795-5901 (fax)



## MODAL FILTER YD METHOD

The “modalFilterYD” method (**Figure 11**) is called as part of the event processing in the window. It is needed to control scrolling of the lower scroll list (the “Select List”). It is written such that it will only handle clicks within the scroll lists vertical control.



**Figure 11.** “modalFilterYD” controls scrolling of the lower scroll list.

## DIALOG HOOK METHOD

The “dialogHook” is the biggest of these three methods. It installs the lower scroll list the first time the method is called. It also handles all clicks in the dialog. So for example when the “Add” button is clicked, the dialog hook method defines what happens in the window.

Instead of going through every case of this method (there are 15 cases), I’ll just provide an overview of what the method is supposed to do:

**First Time Through** — Creates the lower scroll list and installs it in the dialog window.

**Add** — Adds the filename of the selected file to lower list. The file is removed from the upper the list through the file filter function. The FSSpec for the file is added to the File Results object.

**Add All** — The names of all available files in the upper scroll list are added to the lower list. The FSSpecs are added to the File Results object. All files that get successfully added to the File Results are filtered from the upper list display by the file filter method.

**Remove** — Enabled if an item is selected in the lower list. The selected item is removed from the File Results object, removed from the lower list and added to the upper list by the file filter method.

**Remove All** — Enabled if there are any files in lower list. All files in the File Results object are removed, all items in the lower list are removed. The file filter method then allows available files to appear in the upper list.

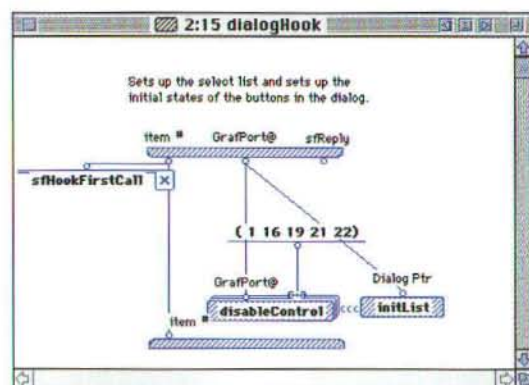
**Cancel** — Deletes all FSSpecs from the File Results object, then closes the dialog.

**Done** — Closes the dialog.

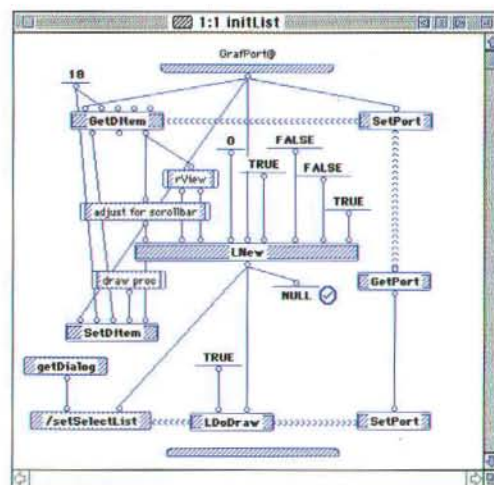
Here is the code of two interesting cases to drive home how things work; what happens the first time though — that’s when we create and install the lower scroll list — and how Add works. I leave the other cases as a review exercise for you and your CPX Debugger.

## First Time Through

The first time “dialogHook” is executed, we must detect that this is the first time and display the lower list box.



**Figure 12.** The First Time Through case creates the lower scroll list item.



**Figure 13.** “initList” creates a new scroll list and places it in the position of window item 18 in the dialog resource.

The Toolbox call “LNew” creates a new scroll list. We define the rectangle for the list by getting the rectangle for window item 18 (see **Figure 9**). We “send” the new scroll list to the MultiFile Dialog in the message “/setSelectList”.



## Add file

The Add file case duplicates the FSSpec representing the selected file and sends that FSSpec as an argument in the call to "addFile".

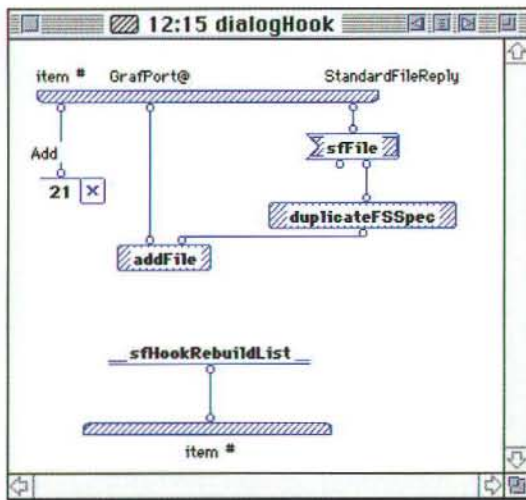


Figure 14. The "Add" case calls "addFile" universal.

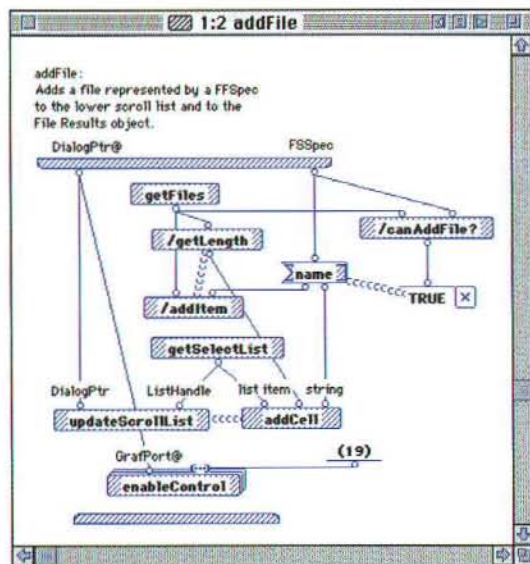


Figure 15. "addFile" universal also enables item 19 in the dialog window.

The "addFile" universal method performs several tasks. It first checks if the file can really be added. If so, it continues on by getting the value of the name field in the FSSpec and asking the File Results object for the number of items in its files list to form an index. It send those two values (name and index) as arguments to "addCell" which adds a new cell to the "Select List". It then sends the FSSpec to the File Results object in a message "/addItem" which adds the FSSpec to the File Result



...let **CameraMan** do the talking.

### THE #1 SCREEN RECORDING & EDITING UTILITY

Fast, simple and easy to use, CameraMan™ captures screen activity and live audio to a QuickTime movie file. For the ridiculously low price of \$69.95 US, create dynamic tutorials, demos and presentations. So don't be shy – download and purchase directly from <http://www.mwg.com> and get your message across with CameraMan.™



MOTION  
WORKS  
GROUP LIMITED

object. After all that, the lower scroll list is redrawn in method "updateScrollList" and window item 19 (Remove All) is enabled.

At this point "dialogHook" case 12:15 (Figure 14) outputs "sfHookRebuildList" and the method ends. The new item now appears in the lower scroll list and disappears from the upper list. The multifile dialog window waits for the next event.

## CONCLUSION

This article describes the CPX code in the MultiFile section. MultiFile contains classes for creating special file dialogs that don't come out of the box with CPX. These classes are useful if you need a file dialog that can deal with several files at once or if you want to specify a unique prompt in a standard file dialog.

With the information in this article, CPX developers can now start using CustomGetFile based file dialogs in their applications. Simply add the MultiFile section to your project, create an instance of MultiFile Dialog, and send it the message "/multiFileGet". All of your problems will be solved.

## BIBLIOGRAPHY AND REFERENCES

Apple Computer's Inside Macintosh: Files, page 3-51.







# OPENSTEP Sockets

## *Network programming in the brave new world of BSD Sockets, a networking interface under Rhapsody*

### WELCOME TO THE PLEASURE DOME

The Macintosh is about to take a large step into a brave new world. It's not quite the brave new world you or I expected or perhaps wanted, but it's going to happen nevertheless. Despite Apple's assurances to the contrary, I expect the first manifestation of the next-generation Macintosh OS to bear more resemblance to NEXTSTEP than to the current MacOS. Hence, it's a fair bet that we'll be getting UNIX-style networking. Indeed, a large clue has appeared with the news that Open Transport is on the list of projects that Apple is no longer continuing to develop.

I've been doing networking stuff for a good while, for MacOS and NEXTSTEP as well as other environments; so, it seemed a good idea (at the time) to write an article explaining how to use the new (actually rather old) APIs to do useful things.

### Assumed Knowledge

The purpose of this article is to describe in detail the BSD sockets API, rather than the ins and outs of TCP/IP. Hence, I'll assume

you've done a fair bit of network hacking before. You should be comfortable with IP addresses and port numbers, binding and connecting, and that kind of thing.

### What's in this article

To start with, I'll describe a few major things that are different between the good old Mac APIs we're used to and the even older BSD sockets APIs you're probably going to have to get used to. I'll then provide a library of routines that make dealing with the socket library a touch easier, and finally I'll use the library to construct a simple server application.

### DIFFERENCES BETWEEN MacOS AND BSD SOCKETS

#### Synchronicity

MacTCP, and to a lesser extent OT, were developed for the MacOS environment, which is essentially co-operatively scheduled. Hence, making synchronous I/O calls is a bad idea. Establishing a connection to a slow or distant site could take a good few seconds, during which time the machine can not respond to any user action. For this reason, these calls normally should be called asynchronously with one of a variety of strategies used to detect and respond to completion. It's beyond the scope of this article to go into those strategies, but suffice it to say that they boil down to completion routine chaining, checking during idle time, or using a Thread Manager thread to spin in a yielding loop.

In complete contrast however, the socket library was initially developed for Berkeley UNIX, a pre-emptively scheduled environment, and therefore synchronous calls are the norm. Indeed, there are no well-developed asynchronous facilities such as you would find under MacOS, because there is generally no need for them. Provided Rhapsody delivers on its pre-emptive multitasking promise, we can all use synchronous calls and get away with it.

**Jason Proctor** is Minister of Ideology for BroadQuest Inc, a startup doing cool things with the network. In past lives he has emulated PCs for Insignia, done the networking for the cool but ill-fated Software Ventures web browser, and worked on reliable multicasting for GlobalCast. Even though he finds himself writing Unix code more often than he'd like, he considers himself a Macintosh bigot.



# KAIDAN

# The Leader in Photographic VR and QuickTime® VR Equipment announces...

## KiWi™

The KiWi™ is our most affordable VR panhead, bringing digital photographic panoramas to an even wider audience. It's the perfect complement to the latest generation of easy-to-use and affordable QuickTime® VR or RealVR™ applications such as, **PictureWorks' Spin™ Panorama, Panimation's Nodester™ & RealSpace's PhotoVista™**.

The KiWi™ consists of two intersecting black anodized aluminum struts that adjust and lock to accommodate the nodal points of a wide range of digital & conventional cameras. When unlocked, the KiWi™ disassembles into two flat pieces that store easily in your camera bag.

Cameras with a tripod mounting thread-to-nodal point distance of 4 3/8 inches (113 mm) and a camera mounting surface-to-nodal point distance of 5 1/4 inches (133 mm) are supported. Scales are provided for both axes, so it's easy to set and record the dimensions for any camera and lens combination.

Mounting your camera to the KiWi™ is easy. Select the appropriate mounting slot on the upright bracket and use the knob to secure the camera. Although the clamping knob is removable and can be placed in any of the three mounting slots, it's a captive fastener so it will stay attached to the unit at all times.

KiWi™ attaches to any standard tripod and camera equipped with a standard 1/4-20 mounting thread. KiWi™ uses a special friction joint that provides smooth & consistent drag while indexing the camera. A label on the base provides a clear indication of the location and increment angle of the camera.

KiWi™. Designed for the professional, priced for the novice. \$99.95 (US SRP)

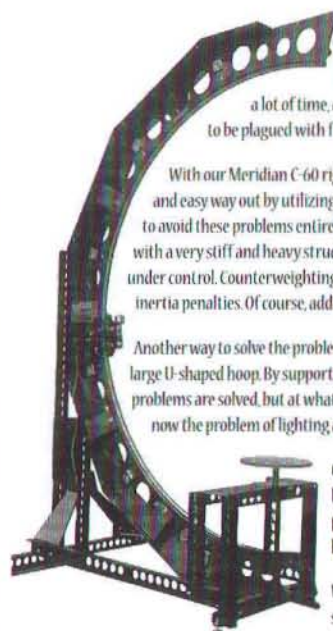


## QuickPan™



Complementing the KiWi™ products, our QuickPan™ line provides additional options and capabilities. Features such as indexed 'click-stop' detents, precision bubble levels, micro tilt level adjustments, a stereo-3D twin camera bracket and an extension tube that expands to 12 feet, for those over-the-crowd shots, make the QuickPan units ideal for the professional VR photographer. In addition, all of our QuickPan models can be used as indexed turntables for single row object movies.

## Meridian™ C-60



Sure, you could design and build your own large, motorized object rig. But whether it's built from two-by-fours, PVC pipe or Army surplus parts, it's still going to take a lot of time, effort and money. And in the end, you're still probably going to be plagued with flexing, vibration and lighting/shadow problems.

With our Meridian C-60 rig, these problems are gone. Since we didn't take the obvious and easy way out by utilizing a swing arm for our large design family, we have been able to avoid these problems entirely. For example, with a large cantilevered arm design, even with a very stiff and heavy structure, it's almost impossible to keep vibration and flexing under control. Counterweighting helps to some degree, but it too adds mass and exacts inertia penalties. Of course, additional mass means more powerful and expensive motors.

Another way to solve the problem, is to avoid the cantilevered arm altogether and to use a large U-shaped hoop. By supporting the structure at both ends, the vibration and flexing problems are solved, but at what cost? The mass and inertia problems are still present, and now the problem of lighting and shadow elimination must be dealt with.

Object movie creation demands precise and flexible lighting setups. Hoop-based rigs simply get in the way of proper lighting and cast unwanted shadows on the object being photographed.

With the Meridian C-60, there are no swing arms or supports to get in the way. You can place lighting wherever it's needed and you can be sure that the structure will not get in the way. The camera rides on a motorized carriage that moves along the inside of a curved 'C' shaped track. With this patented technique, the mass of the camera and drive mechanism stays the same no matter what radius of rotation is used. The only thing that changes is the distance that the carriage must travel along the track. Kaidan intends to provide additional track structures with different radii in the future. It should also be noted that since we're only moving the camera and carriage, we can use a small and affordable motor package.

Meridian™ C-60. Contact us for complete info. Limited time, introductory price: \$999.95 (US SRP)

## Magellan™



Kaidan's line of swing-arm based Object Rigs consists of tabletop to desktop sized machines in manual or motorized configurations. From the affordable Magellan QC (shown at far left) designed to take advantage of the Connectix Color QuickCam, to the fully motorized and software driven Magellan 1500, we're sure to have an object rig that is sized for your task at hand as well as your budget. There's even an upgrade kit available to turn your manual Magellan 1000 into the motorized Magellan 1500. The Magellan products, like all Kaidan products, come with a 30-day money back guarantee and a 1-year warranty.

**Kaidan Incorporated**

703 East Pennsylvania Blvd. Feasterville, PA 19053 · eMail: [info@kaidan.com](mailto:info@kaidan.com) · <http://www.kaidan.com> · Phone: 215-364-1778 · Fax: 215-322-4186



Of course, that's not to say that there aren't any asynchronous facilities at all. Whilst being universally panned for having a dearth of industrial-grade real-time features, UNIX does provide signals as a means of being informed when something happens.

## Portability

Hitherto, Mac developers have not had to consider portability issues when writing MacTCP or Open Transport code. Code written to these APIs is not going much further than MacOS, even though OT is loosely based on the XTI standard.

In contrast again, the socket library has been ported to many different environments, and one should always write socket code to be portable. One never knows when one might want to take one's precious code to UNIX, or anywhere else for that matter. Fortunately however, we only have one major portability concern to worry about — the socket library expects all its IP addresses and port numbers in network (i.e., big-endian) format. By virtue of being initially hosted on the 68k processor, MacOS has its bytes the correct way round, which is why we haven't had to worry about it until now.

So, look out for the convertor routines in the code samples. I've illustrated their use with comments.

## Protocol Independence

The socket library was designed with a certain degree of protocol independence in mind. This manifests in the API as socket calls taking several configuration parameters, and as other calls, if appropriate, taking pointers to generic address structures — pointers to protocol-specific address structures must be cast appropriately.

However, implementing new transport protocols (such as UDP and TCP) generally means writing kernel servers, not a trivial task, and the competing XTI standard (on which OT is based) provides a slightly better interface for alternative protocols, hence other protocols which use the sockets interface are rare.

One extra protocol is normally available under standard BSD sockets, that of UNIX domain sockets. Although certain UNIX systems use these for IPC, they're generally used by normal people about as often as PowerTalk, so I won't reference them further. Details are available in the UNIX manual pages.

## NETWORK API CONCEPTS

The BSD socket library API calls are substantially different to MacTCP and, to a lesser extent, OT, but the actual concepts one deals with are similar, as at the end of the day it's all just TCP/IP. Hence, anyone used to sockets, ports, connections, and the difference between stream-based and datagram-based communication will be off to a flying start. Basically you just do this:

- Open a socket.
- Bind to a port.
- Establish a connection (optional for best-effort datagram-based service).
- Send and receive some data.
- Shut down the connection gracefully.

## MAKING THE TRANSITION

### We're all engineers, aren't we? On to the code.

#### Step 0: Dealing with IP Addresses

As previously indicated, the socket library requires its IP number and port values in network (that is, big-endian) format. It's easy to forget to convert backwards and forwards all the time. Also, the IP address structure is rather arcane, so a convenient way of accessing that structure is a big help. So, I use the following routines to make dealing with the socket address structure a touch more convenient.

#### Listing 1: SocketCalls.c

```

SetAddress
This routine configures an internet address structure from the passed in IP number and
port number parameters.

/* assumption: sizeof(long) == sizeof(IP address) */
/* this is not true for IPv6 */
void
SetAddress (struct sockaddr_in *aAddress,
            unsigned long aIPNumber, unsigned short aPort)
{
    /* set address family */
    aAddress->sin_family = PF_INET;
    /* IP number is assumed to be in network format */
    /* this is because IP numbers are rarely constructed manually */
    /* and are normally obtained via name lookup calls, etc */
    aAddress->sin_addr.S_un.S_addr = aIPNumber;
    /* port number is assumed to be in host format */
    aAddress->sin_port = htons (aPort);
}

GetAddress
This routine configures an internet address structure from the passed in IP number and
port number parameters.

/* assumption: sizeof(long) == sizeof(IP address) */
/* this is not true for IPv6 */
void
GetAddress (struct sockaddr_in *aAddress,
            unsigned long *aIPNumber, unsigned short *aPortNumber)
{
    /* the IP number is returned in network format */
    *aIPNumber = aAddress->sin_addr.S_un.S_addr;
    /* the port number is returned in host format */
    *aPortNumber = ntohs (aAddress->sin_port);
}

```

#### Step 1: Opening A Socket

To do anything, you must open a socket and tell the socket library what kind of communication you're ultimately going to be doing. When writing to the socket library, you call `socket()` and supply the appropriate arguments.

The `socket()` call takes three arguments — the protocol family, the type of communication to be used, and the protocol within the protocol family. If we're intending to communicate TCP/IP, the first argument will always be `PF_INET`, signifying the internet domain protocol family. The second argument will be `SOCK_STREAM` or `SOCK_DGRAM`, according to whether we will be communicating via streams or datagrams. The third argument will be `IPPROTO_TCP` or `IPPROTO_UDP`, according to whether we will be communicating with the TCP or UDP protocol.

Note that the latter two arguments are of course linked. If you want to communicate using TCP, you must specify `SOCK_STREAM` as the second and `IPPROTO_TCP` as the third



argument. For UDP, you must use SOCK\_DGRAM as the second and IPPROTO\_UDP as the third.

Other protocol types are available from the PF\_INET family, such as the ICMP and RAW types, but they are rarely used by clients so I won't go into them here.

#### Listing 1: SocketCalls.c

This routine opens a socket which is configured for stream-based communication over TCP

```
int
OpenTCPSocket (int *aNewSocket)
{
    /* internet address family, stream-based, TCP */
    *aNewSocket = socket (PF_INET, SOCK_STREAM, IPPROTO_TCP);
    if (*aNewSocket == -1)
        return errno;
    else
        return 0;
}
```

This routine opens a socket which is configured for datagram-based communication over UDP

```
int
OpenUDPSocket (int *aNewSocket)
{
    /* internet address family, datagram-based, UDP */
    *aNewSocket = socket (PF_INET, SOCK_DGRAM, IPPROTO_UDP);
    if (*aNewSocket == -1)
        return errno;
    else
        return 0;
}
```

#### Step 2: Binding A Port

Unlike AppleTalk and IPX, TCP/IP exposes port numbers, rather than raw socket numbers. Hence it is necessary to associate our socket with a port number so that we can see and be seen on the network. As is usual with TCP/IP interfaces, we can either specify a port number or ask the socket library to give us the next unused one. Generally, servers would specify a well-known port number so that clients can find them. Under UNIX, and probably Rhapsody, the program would have to be running with root permission to bind to a port in the server range (less than 1024). This is to prevent user processes from masquerading as servers and snarfing people's passwords, amongst other nastiness.

The bind() call takes three parameters — the number of the socket to be bound, the address to which the socket is to be bound, and the size of the address parameter. Note that the address parameter usually has an address member of zero, signifying the local IP address of the machine, however this need not be the case. Multiple IP addresses can be supported, and this is how web servers accomplish virtual hosting.

You'll notice that there's an extra call in the Bind() routine: the setsockopt() call. This call gives the TCP stack permission to bind to the same port each time. Otherwise, the stack imposes a 2 minute delay between binds to the same port. This delay is to guard against the following sequence of events:

- Local endpoint opens a socket and binds to port n.
- Remote endpoint establishes connection with port n.
- Local endpoint disconnects in a disorderly fashion; remote endpoint is not aware connection has been dropped.

PLATFORMS: APPLE SYSTEM 7 • APPLE AUX •

Q: What does it take to  
superior client/server a  
A: A SUPERIOR SE

Java Interface!  
New c-tree 6.7!  
Enhanced Servers!  
Check it out!!  
www.faircom.com

**START** with the  
most advanced client-  
side SDK on the  
market: c-tree® Plus  
at \$895.

- Complete "C" Source code
- ROYALTY FREE (Client Side)
- Multiple supported protocols
- Fast, portable, reliable
- Powerful features like transaction processing
- Win95, NT, and Windows 3.1 ready

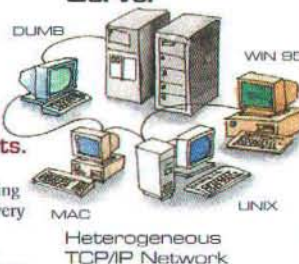
**RESULT**  
A solid, economical,  
easily deployable  
product that fits  
your needs.

- Portable
- Scalable
- Exceptional Performance
- Flexible
- Easy Server distribution
- Convenient OEM terms

**FAIRCOM®  
Server**

**ADD** a strong,  
multi-platform,  
industrial-strength  
Server that supports.

- File mirroring
- Heterogeneous networking
- Automatic disaster recovery
- Multi-threaded design
- Best price/performance available: from \$445- \$3745



You can't find a better client SDK with these features!  
Over sixteen years of proven reliability and performance.  
No one else supports over 30 platforms in this price range!

**c-tree Plus®**

- Complete C Source
- Single/Multi User
- Client/Server (optional)
- Full ISAM functionality
- No Royalties
- Transaction Processing
- Fixed/Variable Length Records
- High Speed Data/Index Caching
- Batch Operations
- File Mirroring
- Multiple Contexts
- Unsurpassed Portability

**FairCom® Server**

- Client/Server Model
- Transaction Processing
- Requires <2MB RAM
- Online Backup
- Disaster Recovery
- Rollback - Forward Resolution
- Client-side "C" Source
- Multi-threading
- Heterogeneous networking
- File Mirroring
- OEM/Source Available

**FOR YOUR NEXT PROJECT CALL FAIRCOM: YOU  
CAN'T FIND A BETTER HETEROGENEOUS  
CLIENT/SERVER SOLUTION!**

Also inquire about these FairCom products:

d-tree™  
r-tree®  
ODBC Driver



**FAIRCOM®  
CORPORATION**

Since 1979

WWWWeb Address: <http://www.faircom.com/>  
800-234-8180

U.S.A. phone (573) 445-6833 fax (573) 445-9698  
EUROPE phone (035) 773-464 fax (035) 773-806  
JAPAN phone (0592) 29-7504 fax (0592) 24-9723

• HP9000 • RS/6000 • SUN O/S 4.X •



# Build great applications... **Better** **Cheaper** **Faster!**

## Tools Plus

LIBRARIES + FRAMEWORK

Make cool apps while others are still reading their manuals.  
Professionally polished. Wickedly fast. Delightfully efficient.

"...the routines are more compact and faster  
than anything you might write... Every element  
of Tools Plus is useful... a bargain compared  
with coding these routines yourself."



You build the interface.

Tools Plus provides the infrastructure.

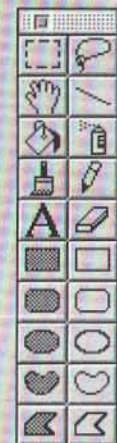
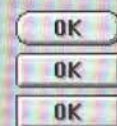
It makes all your pieces work together as an application.

With only a few hundred routines, Tools Plus thins your code  
by tens of thousands of lines. You see results sooner.

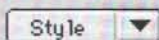
Changes are a snap.

"All in all, it's an incredibly rich collection of tools...  
If you are interested in developing applications  
that have 'quality' written all over them, then  
Tools Plus is for you." **MacTech MAGAZINE**

### Yes, Tools Plus has it!



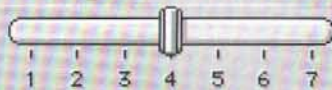
- Create any element using a single routine
- Everything works as soon as you create it
- Automates all standard GUI elements
- Windows, Tool bar and floating palettes
- Buttons (all kinds, flat and 3D)
- Scroll bars (speed control, live scrolling)
- World-class custom controls
- Fields (w/scroll bars, filters, auto-editing)
- Picture buttons (the best anywhere)
- List boxes
- CDEF and LDEF automation
- Cursors (color, animated, auto-change)
- Menus (pull-down, hierarchical, pop-up)
- Edit menu (undo/redo, automatic editing)
- Panels (3D, group boxes, lines, panes)
- 3D titles (raised or inset)
- Extended multi-monitor and color support
- Clipboard automation
- Dynamic alerts (no resources required)
- Event processing automation
- Over 500K of custom fonts, icons, cursors,  
and other useful resources



Plus thousands more exciting features and services!

Results:

- ☒ Radical
- ☐ Slick
- ☒ Ravin'



Inset Panel

Raised Panel

**FREE**

**SuperCDEFs™** \$89 value

- ✓ professionally designed and crafted controls
- ✓ dozens of 3D & flat buttons, tabs, sliders

Tools Plus for

Symantec (THINK) C/C++ (68K)	\$149
THINK Pascal (68K)	\$149
THINK C/C++ & THINK Pascal	\$199
CodeWarrior Bronze (68K)	\$199
CodeWarrior Gold (68K and PowerPC)	\$249

(We accept VISA and Amex only. Add \$10 for shipping.)  
\*Call for Academic pricing

Water's Edge Software  
2441 Lakeshore Road West, Box 70022,  
Oakville, Ontario Canada L6L 6M9

Orders and Enquiries:

Phone: (416) 219-5628

Fax: (905) 847-1638

WaterEdgSW@aol.com

**Free Evaluation Kit:**

Available at our web site

<http://www.interlog.com/~wateredg>

Water's Edge Software

- Local endpoint opens another socket and binds to port n.
- Remote endpoint sends packet to port n, believing previous connection is still up.
- Local endpoint receives packet from previous connection.
- This is bad.

Thankfully, this happens very rarely nowadays.

### Listing 2: SocketCalls.c

This routine associates a port number with a socket which has been opened via the socket() call. If the port number is zero, the next available port number is allocated from dynamic port space (ie outside the server port range).

```
int
Bind (int aSocket, struct sockaddr_in * aLocalAddress)
{
    int    on = 1;

    assert (aLocalAddress);
    /* say it's OK to reuse our address */
    setsockopt (aSocket, SOL_SOCKET, SO_REUSEADDR, &on, sizeof
(on));
    if (bind (aSocket,
              (struct sockaddr *) aLocalAddress,
              sizeof (struct sockaddr_in)) == 0)
    {
        return 0;
    }
    else
    {
        return errno;
    }
}
```

### Step 3: Establishing a Connection

Establishing a connection is mandatory only for reliable, stream-based communication. Within the TCP/IP family of protocols, at least at the network/transport layer, this means TCP. If only best-effort, datagram-based (that is UDP) communication is required, connecting is not required, although if connect() is called on a UDP socket, this fixes the source and destination for further communication. This can be handy if you want to use unicast UDP but don't want to validate that the remote address is correct each time you receive a packet.

Connections are initiated via the connect() call. This takes three parameters — the local socket number (which must be bound to a port), the address of the remote endpoint to connect to, and the size of the address parameter.

### Listing 3: SocketCalls.c

This routine initiates a connection with the remote address. For TCP sockets, this actually negotiates a connection. For UDP sockets, it simply fixes the remote address.

```
int
Connect (int aSocket, struct sockaddr_in * aRemoteAddress)
{
    assert (aRemoteAddress);

    if (connect (aSocket, (struct sockaddr *) aRemoteAddress,
                sizeof (struct sockaddr_in)) == 0)
    {
        return 0;
    }
    else
    {
        return errno;
    }
}
```



Listening for incoming connections is accomplished with the `listen()` call, and accepting with the `accept()` call.

The `listen()` call takes two arguments — the local socket number, which must be bound to a port, and a number signifying the “queue length”. The latter is simply how many incoming connections can be held unaccepted — any more are refused. Generally, you would want to tailor this number according to how many requests you are serving in a given time period.

The `accept()` call takes three arguments — the local socket number, which must be bound to a port, a pointer to an address structure, and a pointer to an integer which is initialized to the size of the address structure. The address structure and size argument are filled in with the address of the endpoint initiating the connection.

Note that the `accept()` call returns a new socket, which is the connected socket over which all data transfer should be done. The original socket is dedicated to waiting for incoming connections, and should not be used for any data transfer. In this way, there is always a socket waiting for an incoming connection, and incoming connect requests cannot be lost.

#### Listing 4: SocketCalls.c

This routine listens for an incoming TCP connection.

Accept

```
int
Accept (int aSocket, struct sockaddr_in *aRemoteAddress,
        int aQueueLength, int *aNewSocket)
{
    int    addressLength;

    assert (aRemoteAddress);
    assert (aNewSocket);

    /* listen() is included here for clarity */
    /* technically, one must only call listen() once */
    if (listen (aSocket, aQueueLength) == 0)
    {
        addressLength = sizeof (struct sockaddr_in);

        *aNewSocket = accept (aSocket,
                              (struct sockaddr *) aRemoteAddress,
                              &addressLength);

        if (*aNewSocket > 0)
            return 0;
    }

    return errno;
}
```

#### Step 4: Sending and Receiving Data

Sending and receiving data can be as straightforward as UNIX file I/O, providing the socket is connected and no special options have to be set on the data to be sent. Normal `UNIX read()` and `write()` work just fine. However, the socket library does of course provide primitives for sending and receiving data in all cases, and clients generally use those.

The calls are split into two groups, for connected and unconnected sockets. For connected sockets, the `send()` and `recv()` calls are essentially the same as `write()` and `read()`, except that an additional flags parameter provides a mechanism for associating special options with the data. It's beyond the scope of this article to go into the options, but UNIX manual pages can provide full details.

For unconnected (that is, UDP) sockets, the calls are similar, but with the addition of separate destination (for `sendto()`) and source (for `recvfrom()`) parameters, and associated address size parameters. You must specify the destination for a send on an unconnected socket, and provide space for the source of a packet received on an unconnected socket.

#### Listing 5: SocketCalls.c

This routine sends the passed-in chunk of data to the other end of the connection. For unconnected UDP sockets, see `SendTo()`.

Send

```
int
Send (int aSocket, void *aData, int aLength)
{
    assert (aData);
    assert (aLength);
    if (send (aSocket, aData, aLength, 0) == aLength)
        return 0;
    else
        return errno;
}
```

This routine sends the passed-in chunk of data to the specified address and port. The socket must be unconnected.

SendTo

```
int
SendTo (int aSocket, void *aData, int aLength,
        struct sockaddr_in *aDestination)
{
    assert (aData);
    assert (aLength);
    assert (aDestination);
    if (sendto (aSocket, aData, aLength, 0,
                (struct sockaddr *) aDestination,
                sizeof (struct sockaddr_in)) == 0)
    {
        return 0;
    }
    else
    {
        return errno;
    }
}
```

This routine receives data from the other end of a connection, up to the amount specified in the parameters. For reception on unconnected UDP sockets, see `ReceiveFrom()`.

Receive

```
int
Receive (int aSocket, void *aData, int *aLength)
{
    int cc;

    assert (aData);
    assert (aLength);
    cc = recv (aSocket, aData, *aLength, 0);
    if (cc >= 0)
    {
        /* note that zero receive generally means orderly shutdown at the other end */
        *aLength = cc;
        return 0;
    }
    else
    {
        *aLength = 0;
        return errno;
    }
}
```



This routine receives data from an unconnected UDP port.

ReceiveFrom

```
int
ReceiveFrom
(int aSocket, void *aData, int *aLength,
 struct sockaddr_in *aSource)
{
    int    cc;
    int    addressLength;

    assert (aData);
    assert (aLength);
    assert (aSource);

    addressLength = sizeof (struct sockaddr_in);

    cc = recvfrom (aSocket, aData, *aLength, 0,
        (struct sockaddr *) aSource, &addressLength);

    if (cc >= 0)
    {
        *aLength = cc;
        return 0;
    }
    else
    {
        *aLength = 0;
        return errno;
    }
}
```

### Step 5: Shutting Down the Connection Gracefully

TCP clients are happiest if they notify one another when a connection is going down. Amongst other benefits, graceful disconnection means fewer packets transmitted on dead connections, which is a good thing for everyone.

Under MacTCP and OT, orderly disconnection is a rather messy process involving closing the local end and then waiting for the remote to close its end. There are a few different situations to survive and most hackers will probably say this is the gnarliest bit of code for these APIs.

Orderly disconnection under sockets however couldn't be easier, largely because the protocol stack handles all the nastiness for you. The `shutdown()` call takes two parameters — the socket to be affected and an integer specifying which ends of the connection to close. If the second parameter is zero, further reception is prevented (that is, the remote end is closed); if it is 1, further transmission is prevented (that is, the local end is closed); if it is 2, both ends are closed and the connection is torn down.

### Listing 6: SocketCalls.c

Disconnect

This routine disconnects both ends of a connected socket.

```
int
Disconnect (int aSocket)
{
    if (shutdown (aSocket, 2) == 0)
        return 0;
    else
        return errno;
}
```

### Domain Name Lookups

Domain name lookups are generally performed by one routine — `gethostbyname()`. This call takes one parameter — a C string (that

is, zero terminated) specifying the name to look up. It returns a pointer to a static structure describing the host, with IP addresses and other names that host might have, amongst other details.

The kicker with the structure returned by `gethostbyname()` is that it contains a list of pointers to IP addresses, rather than copies of them. This is to enable a certain amount of address size independence. Hence one must copy the addresses out of the structure. I've lost count of the number of times I've forgotten this detail and sat there wondering why the addresses I'm getting back are complete crap.

### Listing 7: SocketCalls.c

LookupName

This routine looks up the passed name and places the first matching address in the passed output parameter.

```
int
LookupName
(char *aHostName, struct sockaddr_in *aAddress)
{
    struct hostent *hp;

    assert (aHostName);
    assert (aAddress);

    hp = gethostbyname (aHostName);

    if (hp)
    {
        /* size of address returned is protocol-dependent */
        memcpy (&aAddress->sin_addr, hp->h_addr_list [0],
            hp->h_length);
        return 0;
    }
    else
    {
        return h_errno;
    }
}
```

### BUILDING A DOCUMENT SERVER

This is a simple HTTP-type document server that loops accepting TCP connections on a user-specified port number (defaulted to 8000). When the program successfully accepts an incoming connection, it expects a file name followed by a carriage return or line feed. The server then attempts to open the file and send the contents down the TCP connection. Server-side errors are signified by closing the connection before any content is sent.

### Listing 8: DocumentServer.c

main

Mainline for the document server.

```
#include <errno.h>          /* for errno extern */
#include <fcntl.h>           /* for file open mode */
#include <libc.h>            /* for generic ANSI stuff */
#include <netdb.h>           /* for lookups */
#include <netinet/in.h>      /* for PF_INET stuff */
#include <stdio.h>           /* for printf() */
#include <sys/socket.h>      /* for socket API */

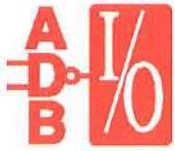
#define kDefaultPortNumber 8000

#define assert(x) if(!(x)) \
    printf("%s.%d: assertion failure\n", __FILE__, __LINE__);

int
main (int argc, char *argv [])
```



# Mac, go out and touch the world for only \$199!



ADB I/O lets your customers' Macs control things, it lets them feel.  
ADB I/O lets the Mac be part of the physical world.

## Thousands of Uses

Science, Multimedia, Children's Museums, Home Automation,  
Theatre Stages, Industrial Testing, Medical Research, Bonsai  
Watering, Robotics, Weather Stations—anything that can

be electronically measured or controlled can use the ADB I/O.

## No Serial Ports Occupied

ADB I/O uses the Apple Desktop Bus to communicate inputs and  
outputs to and from your Macintosh. (Maximum polling frequency  
is 90 Hz.) No external power supply is needed.

## Eight I/O Channels Provided

Four relays for output. Four channels for Digital In,  
Digital Out or 8-bit Analog In.

## Extensive Software Support

With ADB I/O and nearly any environment,\*  
it is easy to build customized  
applications for your control and  
data acquisition needs.

For more info, visit us at  
[www.bzzzzzz.com](http://www.bzzzzzz.com).



\*ADB I/O supports three language environments, two scripting environments, multimedia development environments, and other specific application environments with more on the way. Visit our home page to find out more and to download the complete manual as well as all supporting software at <http://www.bzzzzzz.com>.  
(818) 304-0664 voice. e-mail: [contact@mail.bzzzzzz.com](mailto:contact@mail.bzzzzzz.com) (818) 568-1530 fax. © 1997 BeeHive Technologies, Inc. All rights reserved.



```
char          buffer [1024];
int           acceptingSocket;
int           amountRead;
int           amountReceived;
int           dataSocket;
int           fd;
int           i;
int           portNumber;
int           local;
int           remote;
struct sockaddr_in
struct sockaddr_in

/* make a socket */
if (OpenTCPSocket (&acceptingSocket) != 0)
{
    perror ("socket");
    exit (1);
}

/* allow setting of port number by arguments */
if (argc > 1)
{
    portNumber = atoi (argv [1]);

    if (portNumber == 0)

        portNumber = kDefaultPortNumber;
}
else
{
    portNumber = kDefaultPortNumber;
}

printf ("listening on port %d \n", portNumber);

/* set up our bind address */
/* note the address is zero, for the local address */
/* the port number is expected in host format */
SetAddress (&local, 0, portNumber);
```

```
if (Bind (acceptingSocket, &local) != 0)
{
    perror ("bind");
    exit (1);
}

/* loop accepting incoming connections */
while (Accept(acceptingSocket, &remote, 5, &dataSocket) == 0)
{
    amountReceived = sizeof (buffer);

    if (Receive (dataSocket, buffer, &amountReceived) == 0)
    {
        /* assumption: we receive the file name in one chunk */
        for (i = 0; i < amountReceived; i++)
        {
            if (buffer [i] == '\r' || buffer [i] == '\n')
            {
                buffer [i] = 0;
                break;
            }
        }

        if (i == amountReceived)
        {
            /* we didn't find a CR/LF in the chunk */
        }
        else
        {
            /* open up the file */
            fd = open (buffer, O_RDONLY);

            if (fd == -1)
            {
                /* couldn't open the file */
                perror (buffer);
            }
            else
```



## TCP/IP Scripting Addition

## The Internet Scripting Solution

The **TCP/IP Scripting Addition** allows you to quickly develop Internet client/server applications using AppleScript®. If you want to script with MacTCP™ and Open Transport™, here's your solution!

- ◆ Supports ScriptEditor, FaceSpan™, and HyperCard™
- ◆ Build net-wise WebSTAR™ CGI scripts and NetScape™ CCI scripts
- ◆ Sample scripts include FTP, Gopher, Telnet, Post Office, E-Mail and more
- ◆ Featured on the Apple® Internet Server

**ONLY  
\$49<sup>99</sup>**

Order now through the MacTech Mail Order Store at  
805-494-9797 or other mail order stores

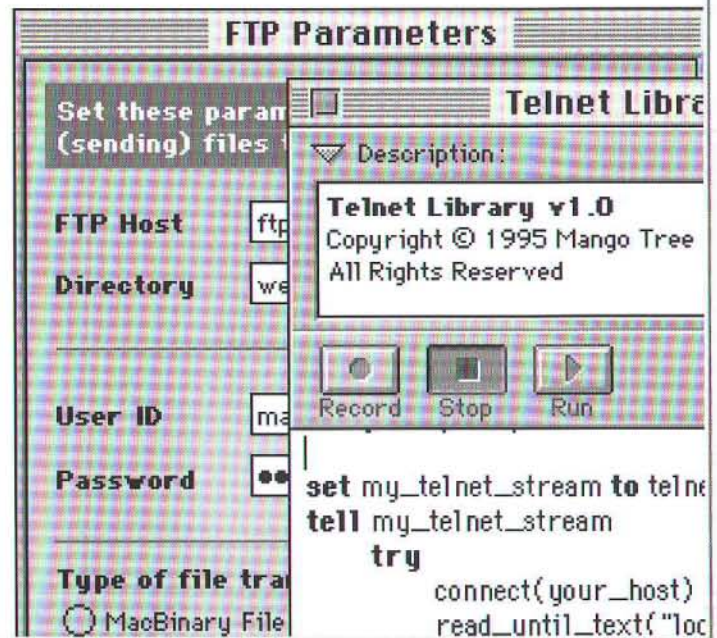


### Mango Tree Software, Inc.

Box 1057 • Brookline, Massachusetts 02146  
617-327-8663

<http://www.mangotree.com/biz/mango>

All trademarks are properties of their respective holders.  
Contact Mango Tree Software for site licensing and redistribution information.  
Copyright © 1996 Mango Tree Software, Inc.



```
do
{
/* read a chunk from the file */
amountRead = read (fd, buffer, sizeof (buffer));

if (amountRead > 0)
{
/* send what we got down the connection */

if (Send
(dataSocket, buffer, amountRead) != 0)
{
/* the initiator has probably closed */
perror ("send");
break;
}
}
while (amountRead == sizeof (buffer));
}
}
else
{
/* couldn't read from the connection */
/* the remote end has probably closed */
perror ("recv");
}

Disconnect (dataSocket);
}


perror ("accept");
exit (1);

/* NOTREACHED */
}
```

### FURTHER READING

Any UNIX manual pages will give you copious information on the socket library. Simply type "man socket" at any available shell prompt. Unfortunately however, the man pages have a tendency to provide amazing detail without actually telling you how to do anything. When I first got into networking, the UNIX manual pages misled me so much that I actually thought AppleTalk was easier to use than sockets. Nothing could be further from the truth.

There are several good books on the socket library in particular and TCP/IP in general. Lots of people learned their TCP/IP from Douglas Comer's seminal work "Internetworking with TCP/IP", in at least three volumes with lots of quality info.

Also worth a look is the Addison-Wesley series "Illustrating TCP/IP", in at least three thicker volumes. These live up to the high standard maintained by A-W's Professional Computing Series. 

**Want to know what products are  
available for MacOS  
development? Check out  
Developer Depot™  
<<http://www.devdepot.com>>**





*by Dave Mark, ©1997 by Metrowerks, Inc., all rights reserved.*

# The CodeWarrior IDE Team

This month's interview is with the folks responsible for the latest rev of the CodeWarrior IDE. The members of the IDE team: Dan Podwall (technical lead), Kevin Bell, Matt Henderson, Glenn Meter, and Rob Vaterlaus.

**Dave:** I'm very excited about the release of version 2.0 of the CodeWarrior IDE. What are some of the new features?

**Dan:** The primary new feature is a ground-up rewrite of the project manager. We gave the old code a decent burial and then started from scratch. We've learned a lot in the four years since CodeWarrior DR1 shipped and had a long list of user requests to address.

The project manager can now build multiple executable files from a single project. This can be variations on a single application, such as 68K and PowerPC versions. Or it could be anything else CodeWarrior supports, such as libraries, code resources, or shared libraries.

A project can also link to other project files, and build those subprojects during make. And of course you can have more than one project open at a time. It's very flexible.

The build system is now threaded and can handle the requirements of C++, Java, and Pascal in a very general way.

We were very concerned about not slowing down builds with all these new

features. In fact, the new IDE is faster than before. In our tests, builds are usually around 30% faster. Initial builds, where the IDE has to search for include files, are up to twice as fast as in 1.7.

The project window now has three different views on the project to accommodate the increased power. There are now optional toolbars in the project, editor, and browser windows, and there's a whizzy new interface for customizing them. Keyboard shortcuts are also customizable via the Key Bindings preference panel.

**Dave:** What impact will the new threaded execution have on the user?

**Dan:** The main benefit is that you can do other things in the IDE while a make is in progress, including editing, browsing, checkin/checkout and multi-file search. So if you get compile errors during a make, you can start fixing them while the rest of your project is still compiling.

We use Apple's Thread Manager to run the build in a separate thread from the user interface. We're planning to thread other features in the future, such as version control and multi-file search. Not only does this make for a more responsive user interface, but we'll be ready to take advantage of modern operating systems like Rhapsody as well as multiprocessing hardware.

**Glenn:** To give some examples of how threads work together, let's look closer at what happens during a build. Under the old IDEs, all compiling and UI happened on the same thread. So, there was not much you could do during a build but wait for it to finish. Under the 2.0 IDE, as Dan pointed out, builds happen on a different thread than the UI. This means that you can still open files, use the browser, perform searches, and even edit files during a build. As the project state is changed during the build, messages are sent from the build thread to the other threads to let them know about the change. When the UI thread next gets a slice of time it will



read its messages and update the project window and browser windows, for example.

**Dave:** With multiple targets per project, will I be able to build fat binaries? How about separate debug and release versions of my program from a single project?

**Glenn:** In the 2.0 IDE, each "target" matches what used to be a 1.7.x "project". Each target includes a file list, segmentation or link order, and preferences to build an application or library. Before 2.0, to build a fat application you would usually have two projects: one to build the 68K app and another that would build the PowerPC code and include the 68K app to leave a fat application. This meant that whenever you needed to update your application, you'd have to keep the two projects in sync. In the 2.0 world, the same thing can be done within a single project. The KillerApp.u project can have one target to build the 68K app and another to build the PowerPC code and create the fat application. Maintaining the project is easier since we can now help you add or remove files from both targets at once. You'll still have to set preferences on a target by target basis, but that will be improved in a later release.

Once you've got related targets in the same project file, you can use target dependencies to make your life even easier. For example, if you have a "Killer Engine" shared library that is used by the Killer application, you can set the application target to depend on the shared library target and to link against its output. Then, whenever the application is built the shared library will automatically be brought up to date. Also, if the name of the shared library is changed, the name will also get changed in the application's file list.

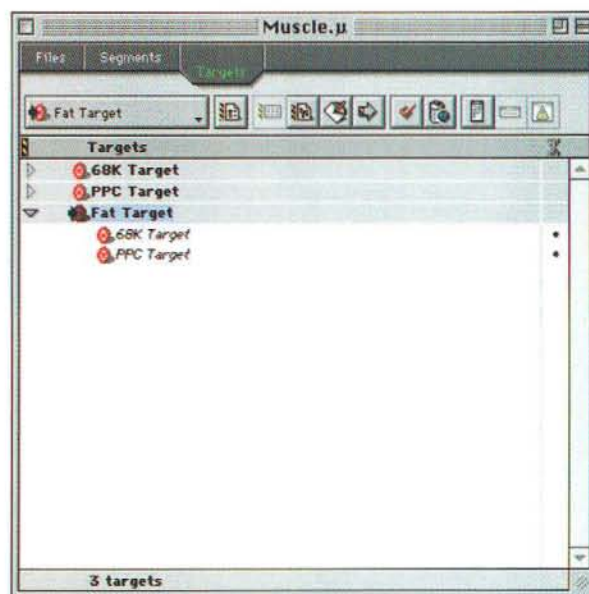
Two new linkers are included in the 2.0 IDE to support working with multiple targets. The "MacOS Merge" linker can be used to merge code and resources into a single file. So, you could also build separate 68K debug and release targets, and use a third target with the merge linker to build the fat application. The other new linker, the "None" linker, actually doesn't generate an output file. It can be used to create "build-only" targets to automate building a set of targets without scripts. For example, you could add a "Build All" target that depends on all other targets and uses the "None" linker. When the "Build All" target is built, all of the other targets will be brought up-to-date without having to leave a dummy output file. This is especially useful when generating groups of applications, libraries, or plugins.

**Matt:** Targets are managed from the targets page in the project window. From the targets page, you can create new targets that are either empty or identical to some other target in the project. Once you've created a new target, double-clicking it will allow you to edit its settings. This is an important difference between CodeWarrior 1.x and 2.0. Language, codegen, and all the other stuff that was configured with

Project Settings dialog in 1.x are now targets settings, so each target in a 2.0 project can be configured however you like.

Building debug, release, 68K, and PowerPC versions of your program is mostly a matter of adding new targets to your project. When you convert a project from CodeWarrior 1.x, it will start out with one target that's configured exactly the same as the project you converted. So if you converted a PowerPC project configured for debugging and wanted to be able to build a release version of your app, you would create a new target using the "Clone existing target" option. This produces a new target identical to the first, so all that would be left for you to do would be to edit the new target's settings so they're configured for shipping instead of debugging. Likewise, to create a 68K target, you just clone an existing target and change the new target to use the 68K linker.

It's easy to set up a target to build a fat app as well. Simply drag a 68K and a PowerPC target under the target that you want to be fat, and a target dependency will be set up so that the 68K and PowerPC targets will be built whenever the fat target is built. You'll also have to turn on the "link output" flag for your 68K and PPC subtargets by clicking in the Link column (which has the chain link icon at top). The "link output" flag tells the IDE to link the output file of a subtarget into the output file of the main target.

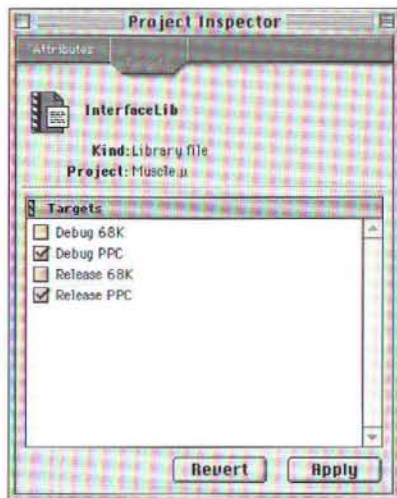


**Figure 1.** The CodeWarrior 2.0 IDE supports multiple targets within the same project.

Aside from different settings, targets can contain different files. Obviously, you wouldn't want to include a PowerPC-specific library in a 68K target. We've introduced a new feature, the Project Inspector, to allow you to examine and change which



targets contain the files in your project. The Inspector always “inspects” whichever files are currently selected in the Files view of the project window. So, to move a PowerPC-specific library out of a 68K target, you would simply select the library, choose the “Project Inspector” command from the Windows menu, uncheck the checkbox for the 68K target, and then press the “Apply” button to save the change. Once the change has been applied, the library will no longer be a part of your 68K target. The Inspector works on all selected items, so if you have several files you need to inspect, use command- or shift-click to select them all at once.



**Figure 2.** The CodeWarrior Project Inspector.

**Dave:** Tell me about nested projects. Can I build a shared library as a subproject within an app that depends on the shared library in order to run?

**Rob:** With 2.0 you can include another project in a target in much the same way you include normal source files in a target. You can select which of the targets of the subproject you want to have built when you build the main target. You can also set whether or not the main target links against the subproject target. You would set this flag when the subproject is generating a shared or static library.

In many cases the same objective can be achieved using either multiple targets or subprojects. Multiple targets are generally preferred when you have different versions of the same code (e.g. PowerPC & 68K, Debug & Release, different localized versions) and you want to build them in various combinations.

Subprojects are nice when you have a library that is shared among many projects and you want to only have to build it once. For example, many of our projects have subprojects for PowerPlant and/or MSL. The subproject has multiple targets for PowerPC & 68K, Debug & Release. Each target in the

main project is set to build and link against the corresponding target in the subproject (e.g. the PPC Debug target in the main project builds and links against the PPC Debug target of the PowerPlant subproject). When changes are made to the library code the subproject will only get rebuilt the first time.

Another situation where subprojects would be preferred over using multiple targets is if you had a product that used a plugin architecture (e.g. Photoshop, AfterDark, or CodeWarrior). You could have one giant project with a target for each plugin and a master build target that had a subtarget for each of the plugin targets. The drawback to structuring the project that way would be that if someone was just doing development on a single plugin they would still need to be working with a project that included the entire source code base which would be unwieldy.

A better way to organize the projects would be to have individual projects for each of the plugins and then have a master project that included all the plugin projects as subprojects. That way it would be more convenient to work on individual plugins, but you could still do a complete build of the entire product in a single make of the master project.

**Dave:** Must I rebuild all my projects to take advantage of this new architecture?

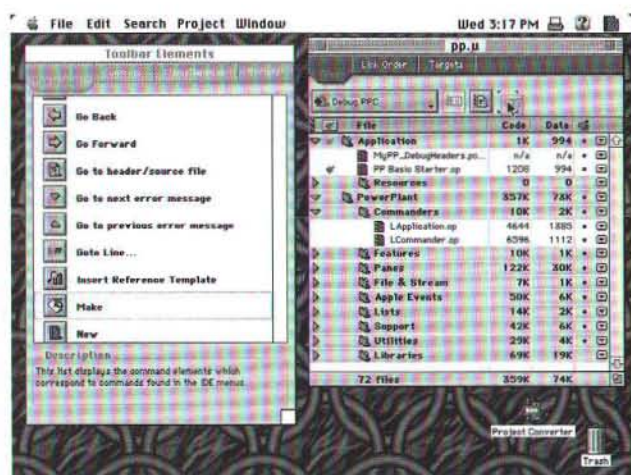
**Kevin:** The 2.0 IDE uses a new project format. You can convert your old projects using the Project Converter that comes with the new IDE. When converting, you have two options: you can merge multiple 1.7 projects into a single multi-target 2.0 project, or convert projects individually. Converting projects individually will create single target 2.0 projects which work the same as the original projects. Merging 1.7 projects allows you to take advantage of new 2.0 features. For example, if you have two 1.7 projects which create 68K and PowerPC versions of your application, you can merge these together with the converter to get one 2.0 project which has 68K and PowerPC targets. If you want to build a fat version of your app, you can create a third target which uses the merge linker to combine the outputs of the 68K and PowerPC targets. To avoid getting lots of duplicate resource warnings you'll probably want to move the resources to a separate target.

**Dave:** What's the story behind the new, configurable toolbars?

**Kevin:** The IDE now has user configurable toolbars in the editor, project, and browser windows in addition to the floating toolbar. You configure the toolbars by dragging command buttons, popups, and other elements from a new element list window. One of the biggest problems I had with the 1.7 toolbar was that you almost had to leave it visible because that's where the IDE reported all status information during builds, file searching, and other operations. In the 2.0 IDE, this



status information is shown in other places so you can hide the toolbars without missing out on important information.



**Figure 3.** The Toolbar Element list, used to configure your toolbars.

**Dave:** Can you tell me about the Visual Source Safe plugin?

**Rob:** When version control is enabled in the IDE, an extra Version Control System (VCS) menu is added to the menu bar. The menu includes commands for all the common VCS commands like Get, Check In, and Check Out. There are recursive variants of all the commands so you can iterate over entire directory trees. The commands can be used from the editor and browser or with the selection in the project window.

The 2.0 IDE supports version control through a plugin interface, so we can support many different version control systems. The plugin API defines abstract VCS operations; a plugin implements these operations for a specific version control system. Our CodeManager product includes a plugin that lets you access Source Safe databases. Several people have come out with shareware plugins that let you use Projector databases and we are working with other VCS vendors to help them develop plugins for their products.

Some of the improvements we've made in the IDE really help out with using source control with project files. In 1.7 the project contains all the object code, dependency info, etc., so you need to make the project file writable just to be able to open and build it. In 2.0 the project file only contains the structural information about the project: the list of source files, subtargets, and settings for each target. All the data generated during a build is stored in a separate per-target data file. This means you can open and build projects without having to check them out. You only need to checkout the project when you add new source files, change the settings, etc.

When you get a new version of a project from the database, if any files have been added or deleted, the IDE will synchronize the target data with the new project file to figure out which files are new and need to be compiled. If the compiler settings haven't changed, you will not need to recompile any of the files that are unchanged from the old version of the project.

Another nice improvement is that we store the VCS settings in a separate per-project settings file so each user can configure VCS with their user name, password, and local root that won't get replaced when they get a new version of the project from the database.

**Dave:** Any other features you'd like to mention?

**Kevin:** The keyboard equivalents for IDE menu commands and editor actions are now customizable via a new preference panel. Any key combinations can be used, and emacs-like prefix keys are supported. Each command can be bound to two keys, so for example, emacs-addicts can bind control-x control-s to save and the standard command-s would still work.

**Matt:** One of the most requested new features is that you can now have multiple projects open at the same time. Perhaps the most obvious change, though, is that the project window has a completely new look — it's now divided into separate pages. The Files page contains the hierarchical list of all the files in the project. This list is for you to organize your files however you want. You can create as many levels of nested groups as you like in the Files page, and the order of files doesn't affect the way your project builds. The Targets page is, of course, where you create and configure the targets in your project. This page has a hierarchical list of your project's targets and their dependencies and subprojects. You'll also see either the Segments page for 68K targets or the Link Order page for PowerPC and Java targets. The Segments page lists the code segments in a 68K target and the files that they contain. This page works like the CodeWarrior 1.x project window. The Link Order page lists all the files in PowerPC target in the order that the files will be linked into your program. This list is flat since PowerPC executables are not segmented.

Also new is that it's now possible to drag things out of the project window. Files dragged out of the project can be dropped into all sorts of cool places. You can drop files onto applications in the Finder, you can drop files into the trash to remove them from the project, you can select a group of files and drop them into the multi-file search list in the Find dialog, or you can drag files from one project and drop them into another. **MT**





by Alan Weissman, TopSoft, Inc.

## Filter It!

### *Join the Plug-in Revolution: Write a FilterTop Filter*

#### THE PLUG-IN REVOLUTION

In 1992 a fruitful collaboration began among a group of Mac programmers on Usenet. Fired up by the idea of pooling their expertise to develop a killer app that would incorporate many of the new features of System 7, these enthusiasts soon formally organized as TopSoft, Inc., and dubbed their chief project FilterTop. Its main purpose: to bring batch processing and pipelining from UNIX to the Mac, using such advanced features as Apple events, drag-and-drop and multithreading.

Four years passed before FilterTop's first full public release. A lot happened in that time. Popular programs had evolved into resource-hungry monsters that had to incorporate every conceivable feature, and this tendency was stifling both programmers and users. To let in fresh air, a contrary trend in Mac programming was emerging, toward modularity and plug-in architecture. We see this trend gaining momentum today as large applications increasingly rely on small, interchangeable plug-ins to provide much of their functionality. Adobe Photoshop is the best-known of these, but others are rapidly following suit.

Well, FilterTop was ahead of the pack, and today, in an age when software bloat and creeping featuritis still dominate, FilterTop seems more innovative than ever. In fact, practically speaking, FilterTop is just a framework for plug-in modules.

FilterTop offers a distinct advantage in the way it uses plug-ins. Each module, or filter, can be linked with others in a pipeline. Then one or a batch of files can be sent down the pipe, with each filter manipulating the files in a specified way. Instead of one filter at a time operating on one file, you have many filters operating in quick succession on many files, each of which might have been created by a different application. The FilterTop framework provides the user interface and the glue that ensures everything — batches of files and assemblages of filters — works together smoothly.

FilterTop shares an exciting property with other applications that have a plug-in architecture: extensibility. A new capability may be added to those already present simply by writing a new filter and plugging it in. The filter programmer doesn't have to know a thing about the inner workings of FilterTop. Even preexisting filters provide a field for experimentation. Different permutations and combinations may provide new functionality to meet specific needs without your writing one additional line of code.

Best of all is that filters are easy to write. Most of the remainder of this article will outline the simple steps to create your own FilterTop filter. First, let's get a general overview of FilterTop's operation from the user's point of view so you can form a better idea of how a filter fits into the functioning of the whole.

#### FILTERTOP IN ACTION

The building blocks of FilterTop are filters. The user can use just one at a time or pipe together two or more. In all cases, however, they must be combined and set up by means of *SuperFilters*. These in turn may be saved in files called *Toplets*, stand alone applets that perform multiple operations on files — not by themselves, but by using Apple events to pass instructions to FilterTop.

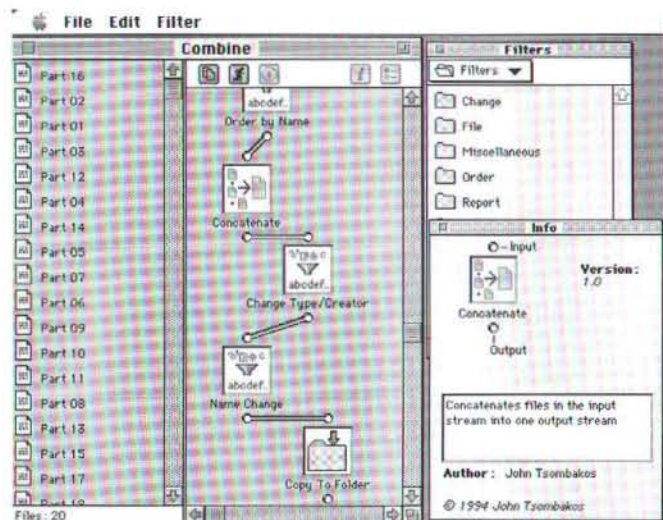
---

**Alan Weissman** believes that the tinkering of hobbyists is as important as the effort of professional developers in keeping the Mac an exciting platform. When not doing his own tinkering, he may be found working in Macintosh tech support, reading detective novels or listening to old jazz recordings, among multitudinous other activities. He may be reached at [alanw@bway.net](mailto:alanw@bway.net).



A FilterTop Toplet is somewhat analogous to an AppleScript droplet; you drop onto its icon in the Finder other icons representing any number of files (or you can just double-click it and add files later). A Toplet is opened, causing a SuperFilter to be recreated according to choices previously made by the user. FilterTop is launched if it is not already running and sent the information that enables it to display icons for the files to be processed and for filters that represent the operations to be performed on the files. If the option to do so has been chosen, the files are automatically processed; otherwise, FilterTop waits for the processing to be initiated by the user.

To create a SuperFilter in the first place, you drag icons from a list (displayed in a floating palette) to a window. Each filter has one or more ports, indicated on the top and bottom of the icon. You use the mouse to connect the filters in custom order by dragging from one port to another. When this is done, the filters are shown connected by pictures of pipes. Thus the concept of pipelining — an abstract one in operating systems without a GUI — is graphically illustrated on the Mac. In fact, with its menus, lists, windows and icons, FilterTop is very much a Macintosh application.



**Figure 1.** A SuperFilter set up and ready to run.

Icons for files to be processed appear in another part of the window. **Figure 1** shows an example. Say you have a group of twenty files. Each is a SimpleText file and contains a section of an article. The files are named "Part 01," "Part 02," etc. Your object is to create a single plain-text file that contains the entire article and will open in Microsoft Word. By dragging and dropping with the mouse, you create a pipeline of several filters in the SuperFilter editor window. The first assures that the files are sorted by their names; the next concatenates the text of all the files; the next changes the creator code of the output file to that of Word; the next assigns a name to the output file; and the final filter sends the result to a folder you have designated.

You click a button and witness the start of processing: the filter icons become dark as data flows through them, the pipes swell when the data flows down to the next filter, and so on. At

least that is the GUI representation of what is going on behind the scenes. Bruce Bennett, an amused commentator on the Internet, called FilterTop "UNIX batch processing by Rube Goldberg." That is the way it looks. (It also shows that members of the Mac community have a sense of humor!) Though FilterTop may appear to be an ungainly contraption that you watch with a smile, it works, and with surprising speed and efficiency.

### ADVANTAGES OF WRITING A FILTER

Writing a filter to perform a given function is easier by an order of magnitude than creating a stand-alone application. FilterTop handles the tough stuff. It coordinates the operations of the various filters and assembles the files passed to it by the user. It passes to your filter the data to be processed. It allows you to retrieve the user's preferences as well as a certain amount of data to be used in performing the operation, such as a string to be searched for. When you are through processing — "filtering" — the data, the FilterTop engine receives the results, which it passes on to the next filter or saves to disk as a file.

Thus, you the filter programmer need be concerned only with the specific task you want your filter to accomplish. This may be simple, such as counting the characters in an input file, or much more advanced, such as applying a sophisticated compression algorithm or translating from one graphics format to another. Professional developers can find intriguing potential in FilterTop filters, yet programming one is simple enough for hobbyists.

### WRITING YOUR FILTER

#### Getting Started

For your filter to work properly you are required to follow a strict set of guidelines. This may seem constraining at first but this system has its advantages. First of all, FilterTop provides programmers with numerous aids to help you comply with its restrictions and get your code working. Secondly, once you set up everything correctly for interaction with the FilterTop engine, you are left with a great deal of latitude in what you can do in the body of your code. In addition, you will find that, once you are comfortable using the framework FilterTop provides, you are freed from the need to spend time on the details of pre- and post-processing, including handling most of the user interface and all but the simplest aspects of error trapping and reporting.

#### The Basic Steps

Here are the five basic steps to create a fully functioning FilterTop filter:

- Receive messages from the engine.
- Retrieve data from the engine.
- Process the data.
- Return the results to the engine.
- Send a return code back to the engine indicating the outcome of the operation.

There are also a few things to be done with resources at different stages. At this time support is provided for writing filters in "C," using either Metrowerks CodeWarrior or THINK C.



Additional support may be added in the future. (It should also be mentioned that FilterTop is not currently PPC native; this will probably change with the next major revision.)

## Receive Messages

To receive messages from FilterTop, you must `#include "FTFilter.h"`, a header file that defines these messages. `FTFilter.h`, supplied by TopSoft with its other programming tools, also defines the data structures and functions that FilterTop uses and with which you must become familiar.

In fact, messages and everything else transmitted to you by FilterTop come through a parameter block, a pointer to which is passed to your filters `main()` function when its code is executed. In this respect as well as in others, as we'll see, a FilterTop filter resembles a HyperCard XCMD. If you have any familiarity with the latter you can easily understand the former. (Photoshop filters work in the same general way; if you can write one of those, a FilterTop filter is a cinch.) The structure of this parameter block is worth studying; once you get the hang of how it works you will have gone a good way toward comprehending the interface between your own code and FilterTop.

```
typedef struct
{
    FilterMessage  serviceRequested;    // essential
    long          filterShellVersion;  // ignore
    Callbacks     *cb;                // essential
    DialogPtr     dlg;                 // rarely used
    EventRecord    *event;             // rarely used
    Handle         filterGlobalsH;      // useful
    Handle         filterConfigH;       // rarely used
    long          data;                // useful
} PBCallToFilter, *PBCallToFilterPtr;
```

Understanding the FilterTop parameter block is easier than it at first appears, because it is essential to deal with only two of its elements. Two others are also useful at times, and the rest are either reserved by FilterTop or used only with certain special types of filters. `filterShellVersion` is reserved by FilterTop. `filterConfigH`, `dlg` and `*event` are used in filters only if the filters must interact with the user through their own self-created dialog. This is rarely if ever necessary since FilterTop also provides a remarkably complete means of putting up such a dialog and retrieving preferences automatically, through a mechanism called *AutoConfig*. More about *AutoConfig* later.

All filters are *reentrant*; that is, if a filter has been called, it may be called a second time (and a third, etc.) before the first instance returns. This imposes the restriction that filters may not have global variables (otherwise, more than one instance of a filter could struggle over the globals). It is permissible, however, for memory allocated on the heap to be used as though it were global, and the member `filterGlobalsH` is provided as a convenient way of accessing a block of these "globals." The member `data` is similar to the `refCon` provided in Toolbox window and control records. Basically, you can put any four-byte value you want there for convenient later retrieval.

The absolutely essential members of the parameter block, however, are the first and the third. (We will come back to the third a little later on.) `serviceRequested` is a variable that you must

## HIGH PERFORMANCE IMAGING AND ANNOTATION DEVELOPMENT TOOLS

**RasterMaster 6.0** - Familiar API and well-proven technology in use world-wide by companies such as British Petroleum, Chase Manhattan, Eastman Kodak, FileNet, Ford, Gannett, Hewlett-Packard, IMSI, Lexis, Philips, Polaroid, Siemens-Nixdorf, Trix, Unisys, and Xerox.

**50+ Raster Formats:** All TIFF, JPEG, Group3, Group4, MO:DCA, IOCA, CALS, PNG, PCX, BMP, GIF, PhotoCD, ABIC, **Flashpix** and more!

**RasterNote™** annotation/redlining toolkit - includes all popular features. Create Sticky Notes, Freehand Drawing, Lines, Ellipses, Polygons.

**NEW ADDITIONS: RasterNote, UNIX, Alpha, Flashpix**  
Format Reading, Saving, Compression, Display, Despeckle  
Rotate, Zoom, Pan, Scroll, Deskew, Anti aliased display,  
Auto aspect ratio, Image processing, Transparent color  
Improved Twain scanning, Color reduction and promotion

**Platforms:** Win 95, Win NT, Win 3.1, OS/2, Alpha, Macintosh, UNIX  
**Environments:** DLL, VBX, OXC/ActiveX, Delphi, VB, PBuilder, VC++

Call 617 630-9495 or see our Website for **FREE, NO RISK EVALUATIONS!**  
[www.snowbnd.com](http://www.snowbnd.com)

PO Box 520 Newton, MA 02159 Tel: 617 630-9495 Fax: 617 630-0210  
Email: [salesmc@snowbnd.com](mailto:salesmc@snowbnd.com)

**SNOWBOUND**  
**SOFTWARE...**  
*Imaging Software - Powerful, Fast, Reliable*

examine immediately whenever your filter is called. It is standard practice simply to set up a switch statement to handle the messages in `serviceRequested`, and in most filters the `main` function will consist of little more than this switch statement.

```
pascal FilterResult main( PBCallToFilter *pb )
{
    FilterResult rc;

    ENTER_FILTER;

    switch (pb->serviceRequested)
    {
        case msgOpenFilter:
            rc = filterMsgNotHandled;
            break;
        case msgFilterData:
            rc = MyFilterFunction( pb );
            break;
        case msgCloseFilter:
            rc = filterMsgNotHandled;
            break;
        default:
            rc = filterMsgNotHandled;
    }
    EXIT_FILTER(rc);
}
```

(`ENTER_FILTER` and `EXIT_FILTER` are macros that enable your filter to use the A4 register to handle static data that the compiler treats as globals.) `msgOpenFilter` and `msgCloseFilter` are occasionally useful if you want to set up some structure or perform a particular action only once at the beginning of a



# MkLinux

Microkernel Linux for the Power Macintosh

**Why wait for Rhapsody?** MkLinux is a complete system, based on Linux 2 and the Mach 3 microkernel. It includes a complete software development system (gcc, gdb, perl, ...), X11R6.3, and hundreds of other commands. Get MkLinux and start working right now with Mach, Objective-C, UNIX development tools, and more...

MkLinux builds and runs most Intel-based Linux software. It works efficiently and reliably on a wide range of Power Macintosh platforms, with other ports on the way. MkLinux has an active user community and a substantial Reference Release, both sponsored by Apple Computer. If you want to get a jump start on Rhapsody, this is a great way to do it!

Visit Apple's MkLinux web site ([www.mklinux.apple.com](http://www.mklinux.apple.com)) and Prime Time Freeware's web site ([www.ptf.com](http://www.ptf.com)) to find out more about MkLinux and other fine freeware products.

Prime Time Freeware  
370 Altair Way, #150  
Sunnyvale, CA 94086

info@ptf.com  
(408) 433-9662  
(408) 433-0727 fax

filtration session and not every time data is sent to your filter. There are other messages that you must know for special types of filters. In most cases, however, the only message you must act on is `msgFilterData`, which simply is the go-ahead that signals you to perform the action that your filter is supposed to perform.

The variable `rc`, of type `FilterResult`, receives a return code. Normally it will be `filterOK`, which indicates that your filter successfully did what it is supposed to do, or `filterMsgNotHandled`, as seen here (which simply indicates that you take no action on this particular message). On occasion you might also return `filterAborted`, in case you were informed by the engine that the user has aborted the filtration, or `filterProcessingError`, to inform the engine (or confirm that you have been informed) that an unrecoverable processing error occurred. Either of these codes, or `filterOK`, might be assigned to `rc` through `MyFilterFunction` in the example above, depending on what happened during filtration. That is all you must know about the `FilterTop` messages.

## Retrieve the Data

The second member of the `FilterTop` parameter block, `*cb`, is extremely important. It is, simply, a pointer to another structure, this one consisting of (at this writing) twenty-two pointers to functions. These functions are the `FilterTop` *callbacks*, a term that, again, you will be familiar with if you have written any HyperCard XCMDs. The callbacks are functions that `FilterTop` provides for interacting with it; the engine passes you

pointers to the callbacks, and you *call back* to the engine by simply dereferencing one of the pointers, passing the parameters required by that particular routine.

```
static FilterResult MyFilterFunction( PBCallToFilter *pb )
{
    FTErr err, writeErr;
    Stream inputStream, outputStream;
    FInfo fndrinfo;
    FSSpec suggested_spec;

    err = pb->cb->Open(input1, &inputStream, &fndrinfo,
                      &suggested_spec);
    HANDLE_ERR(err);
```

Note the double use of the pointer-to-member operator to call `Open`, a pointer to which is passed as part of `cb`, the `Callbacks` structure, which in turn is pointed to by `pb`, the parameter-block pointer. The `Open` callback will become very familiar to you, as it is used to open both input and output streams.

`FilterTop` is built around the model of *streams*. This concept should be familiar to anyone who programs in C. Streams, which are simply sequences of bytes, interpose a layer of abstraction between your filter and the files input to it. Instead of handling the details of file I/O, you simply *open*, *read* from, *write* to and *close* streams.

After opening an input stream, we must open an output stream:

```
err = pb->cb->Open( output1, &outputStream, &fndrinfo,
&suggested_spec );
HANDLE_ERR(err);
```

`input1` and `output1` are constants representing pipes from which to read, or to which to write, the stream in question. `&inputStream` and `&outputStream` are pointers to variables of type `Stream`; the `FilterTop` engine stores in these variables stream identifiers that you need to pass back to it whenever you access the streams. The next two parameters similarly receive Finder information (`FInfo`) and a file-specification record (`FSSpec`) associated with the stream. Usually these have been derived from a file on disk that was the origin of the stream. But since streams do not necessarily correspond to files on disk (they may have been created in a previous filter, for example), this information should not be considered as necessarily bound to a file. It is there so that if needed the engine can use it to create a new file on disk at the end of the filtration process. In most cases, you just pass along what you receive, although specialized filters may alter this information.

`HANDLE_ERR` is a macro that represents a simple error-handling mechanism, defined as:

```
#define HANDLE_ERR( err ) if( err != ftOK && \
                           err != ftEndOfData ) \
                           if( err == ftAbort ) \
                           return filterAborted; \
                           else \
                           return filterProcessingError;
```



## Process the Data

Now you are ready to have your filter perform its filtration.

```
#define BUFSIZE 1024

long bufsize = BUFSIZE;
char buffer[BUFSIZE];

while( err != ftEndOfData )
{
    err = pb->cb->Read( inputStream, buffer, &bufsize );
    HANDLE_ERR( err );
}
```

Use `Read` to obtain a convenient number of bytes from the input stream into a buffer that you have set up. This buffer may hold anywhere from a single byte to several megabytes. The latter is an extreme case. If you want to create a large buffer, you can allocate memory on FilterTop's heap. This may be necessary with certain kinds of filters. But whenever possible use a small buffer on the stack and process your data in small chunks, writing the processed data to the output stream as you go. This enables the data to be passed to the next filter immediately for efficient pipelining. Also remember that available memory must be shared among all currently operating filters, and there may be quite a few of them. FilterTop is multithreaded, and several threads may be active at the same time.

```
/*
Our filter performs a very simple operation: converting all plain lowercase characters
to uppercase; we loop until all characters in the input stream have been read, checked,
converted if necessary and written to the output stream.
*/
```

```
long i;

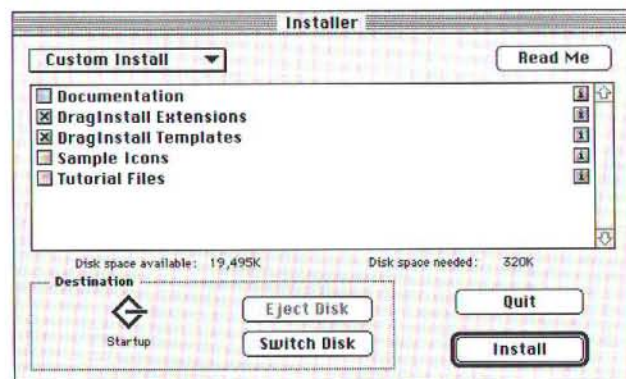
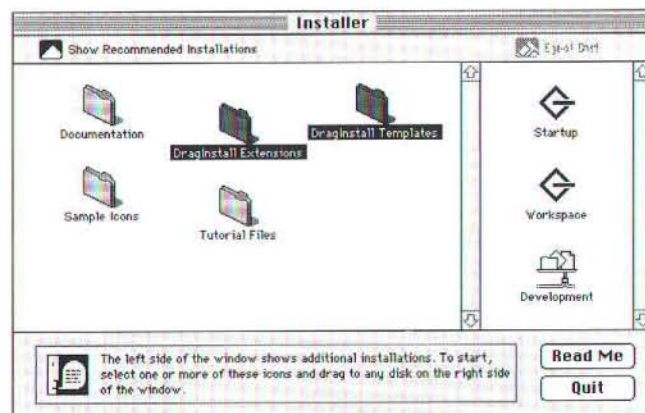
for( i = 0; i < bufsize; ++i )
{
    if( buffer[i] >= 'a' && buffer[i] <= 'z' )
        buffer[i] -= 32;
}
if( bufsize > 0 )
{
    writeErr = pb->cb->Write(outputStream, buffer, bufsize );
    HANDLE_ERR( writeErr );
}
// end while ( err != ftEndOfData )
```

When the `Read` callback returns the code `ftEndOfData`, that is the signal to end the while loop, as all the bytes have been read from the input stream. The `FileInfo` and `FSSpec` were already copied to the output stream when it was opened. There remain only the resources associated with the stream to be copied to the output stream.

```
err = pb->cb->CopyResources( inputStream, outputStream );
HANDLE_ERR( err );
```

`CopyResources` simply copies all the resources at once, and very quickly; if there are no resources it does nothing, so it can't hurt to use this callback if you are not sure about the resource forks of the files you are filtering. FilterTop also provides callbacks for copying individual resources if you should want to do so.

# Guess which installer was built by DragInstall®



## Guess again.

If you guessed that the first installer was built by DragInstall, you're only half right. Now with DragInstall 3.0, you can build *both* of these installers from a *single* script file!

For more information about DragInstall and a free demo, visit our web site at <http://www.sauers.com/draginstall> or give us a call at 1-800-890-9880.



## Return Your Results

Now we close both the input stream and the output stream:

```
err = pb->cb->Close( inputStream );
HANDLE_ERR( err );
err = pb->cb->Close( outputStream );
HANDLE_ERR( err );
```

Notice that we always check for errors. This not only allows us to return control gracefully to the FilterTop engine in case a serious error has been detected (which we can do also if we encountered such an error in our own code; FilterTop offers yet other, more sophisticated means of error reporting that you should look into), it also gives you a way to find out if the user has decided to abort the operation. If the engine returns the code `ftAbort` from any of its callbacks, that is the signal to clean up and immediately return control to FilterTop. With a simple filter like this one, the macro `HANDLE_ERR` is all you need if you receive this message. All it does is return control to FilterTop (with the appropriate return code), and that is sufficient. (Though just barely. Unless your filter is of the simplest kind, you should consider using the `ReportError` callback before returning `filterProcessingError`. There you can pass a message string to the engine indicating the type of error that occurred. The engine then displays this message to the user.) More complicated filters may perform one or two cleanup operations. If any memory was allocated on the heap, it should be disposed of. In most cases, that is all you must worry about.

## Send a Return Code

And, finally, since if we have reached this point all has gone satisfactorily, return to FilterTop (through our main function) with the code `filterOK`.

```
return filterOK;
}
```

Always remember to return a return code whenever you hand control back to FilterTop. If you are responding to close messages, you also must clean up when you receive one. You might, for example, have allocated some quasi-global memory when you were opened; naturally, you must free this when told to close. The FilterTop parameter block includes the field `filterGlobalsH` as a convenient way of holding and managing a handle to a block on the heap; it does not have to hold "globals" but can be used any way you want.

## AutoConfig

Now (continuing with our example) suppose you decide that the user should have the option of deciding whether to convert uppercase to lowercase text or the reverse. You want to display a dialog box giving the user the option to check "Uppercase to Lowercase." In a full-scale application this is relatively simple. Even so, you have to deal with setting up the dialog and trapping events to find out what the user did in the dialog. You could instead add a menu item, but here you are a stand-alone code module, and, running in the context of FilterTop, you cannot provide your own menu item or even trap events, or you might interfere with FilterTop's own operation.

Not to worry. FilterTop provides you, the filter programmer, with all the machinery to retrieve user preferences without having to be concerned about displaying dialogs (except in ResEdit), creating menu items, or handling a single event. This machinery is called *AutoConfig*. With AutoConfig, instead of devoting twenty-five percent of your code to creating and disposing of dialogs and managing event loops, you can handle the whole business of filter configuration and interacting with the user by a few minutes' work with ResEdit or another resource editor, and just a few extra lines of code.

The FilterTop developers' kit provides ResEdit templates for the two special resources required, 'FCFG' and 'VLD'. In addition, you must create one each of the usual 'DLOG' and 'DITL' resources. FilterTop expects all these resources to have the ID of 128. There are three standard items that your dialog must have to be consistent with all filter-configuration dialog boxes. These are the three buttons that the user clicks to "Use Now" (the default, always item 1), "Cancel" (item 2) or "Make Default" (item 3), which tells FilterTop to save the current configuration as the default setup whenever this filter is used.

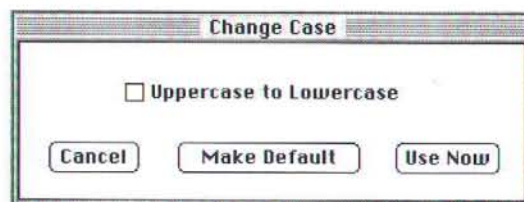


Figure 2. The AutoConfig dialog as seen in ResEdit.

We will add a single item of our own: a checkbox (item 4) with the text "Uppercase to Lowercase". (Figure 2 shows the result of setting up the 'DLOG' and 'DITL' resources.) By convention FilterTop passes on to the filter code the Boolean value of a checkbox with a single character (*not* the actual number), '1' for "true" and '0' for "false." This result is linked with the internal name of the item. This name (which has nothing to do with the name of any resource) is assigned by you in the 'fCFG' resource (Figure 3) and must be a four-character name of type `OSType` — the same kind of name used for resource types, file types and creator codes, among other things. We will assign the name 'UtoL'. A default value can be supplied, which we will make '0'. (We don't actually use the single quotation marks in the resource.)

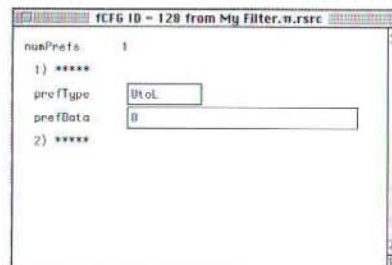
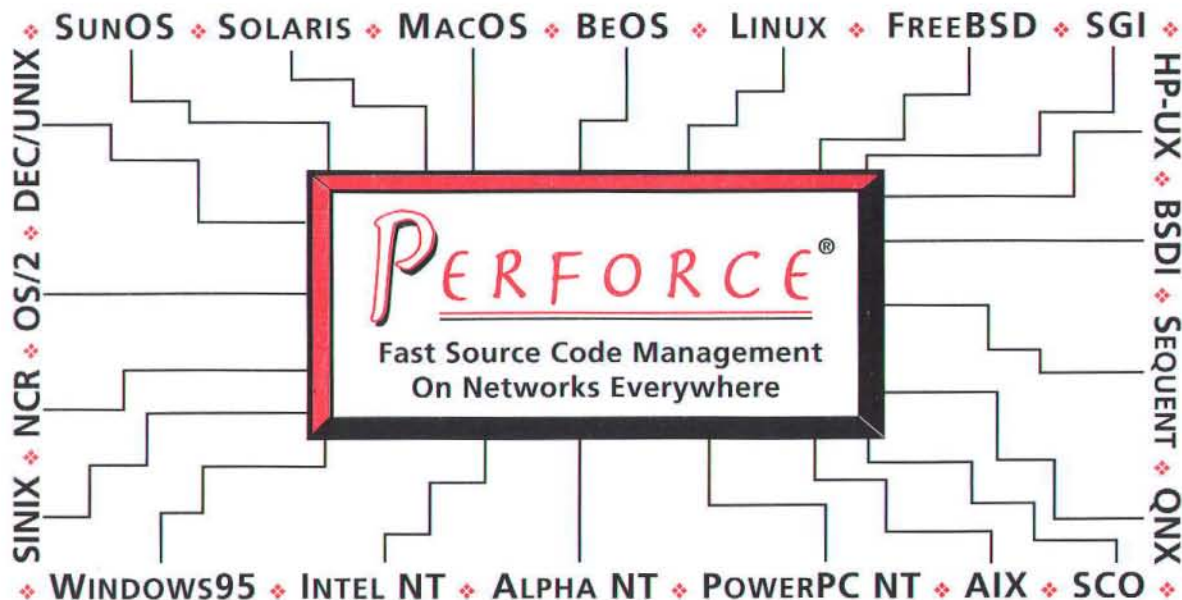


Figure 3. The 'fCFG' resource.





... Find us fast... on MacOS, BeOS and everywhere else... f



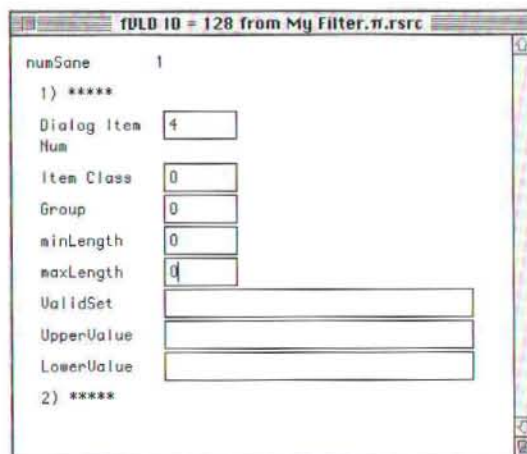
**PERFORCE SOFTWARE, INC.**

1.510.865.8720

[HTTP://WWW.PERFORCE.COM](http://www.perforce.com)

[INFO@PERFORCE.COM](mailto:info@perforce.com)

The values in one more resource must be filled in to set up our AutoConfig dialog: those in the 'fVLD' resource (**Figure 4**). There are several fields to be completed in the template. Values such as "Group," "Class," etc., may be used to link dialog items in complex ways. Also, for dialogs with edit-text items, minimum and maximum values and lengths, as well as a set of valid characters, may be provided. All these options are explained in the filter programmer's documentation. For our simple example, as for all simple checkboxes unlinked to others, we can enter either zero or nothing at all in all the fields. Also be sure to enter the number of the dialog item. (Do *not* create items in the 'fVLD' resource for the three required button items 1 to 3; these are handled automatically.)



**Figure 4.** The 'fVLD' resource.

**Want to suggest an article for the magazine? Send your suggestion to**  
**<<mailto:editorial@mactech.com>>**

This done, modifying our code to retrieve the user's configuration preference requires just a few lines, as follows:

```
char configChar;
long configLen = sizeof( char );

err = pb->cb->CfgFindData('UtoL', &configChar, &configLen);
HANDLE_ERR(err);
```



Then we modify our loop to extend our functionality based on the additional preference:

```
if( configChar == '1' ) //Uppercase to lowercase
{
    for( i = 0; i < bufsize; ++i )
    {
        if( buffer[i] >= 'A' && buffer[i] <= 'Z' )
            buffer[i] += 32;
    }
}
else // Lowercase to uppercase
{
    for( i = 0; i < bufsize; ++i )
    {
        if( buffer[i] >= 'a' && buffer[i] <= 'z' )
            buffer[i] -= 32;
    }
}
```

All this takes longer to explain than to do! (Note: There already exists a "Change Case" filter, written several years ago by TopSoft's founder, Stephen Jovanovic. My example code is slightly different for demonstration purposes but does pretty much the same thing.)

### The 'fINF' resource

At this point we *almost* have a fully functioning filter. We need only to register some information with FilterTop so it will know how many ports we have and what names should be attached to them, and a few other items. This is done in the 'fINF' resource (again, ID 128), which must be part of all FilterTop filters.

Filter Version	100
Min FT Version	100
Cur FT Version	100
Stack Memory Req'd	0
Config Possible	<input checked="" type="radio"/> True <input type="radio"/> False
Config Required	<input type="radio"/> True <input checked="" type="radio"/> False
Auto Config	<input checked="" type="radio"/> True <input type="radio"/> False
Uses Error Port	<input type="radio"/> True <input checked="" type="radio"/> False
Filter Name	My Filter
Author	John Doe
Copyright	©1996

Figure 5. The 'fINF' resource.

Let's quickly run through the 'fINF' items. (See **Figure 5.**) The item "Filter Version" is your own three-digit number. There is an imaginary decimal point between the first and second digits. "Min FilterTop Version" at this time should just be "100." In the future there conceivably could be filters that will run with FilterTop version 2.0 but not earlier, for example. "Cur FilterTop Version" is also now "100," but will naturally change. "Stack Memory Required" must be filled in only if you determine that you will need an unusual amount of stack space. In most cases just leave this at "0" for the default. (It can't hurt to put in an estimate; future versions of FilterTop may require one.) "Config Possible" should be true if you use AutoConfig or are self-configuring and false if not. ("Self-configuration" is an

option that admittedly has not been well supported; this is understandable, however, since AutoConfig has so far provided for all configuration needs anybody has had.) "Config Required" is not currently supported. Set this to false. "AutoConfig" is self-explanatory. "Uses Error Port": FilterTop provides the option of having a special output port for reporting of non-fatal errors — set this to true if you use this option. "Filter Name," "Author," "Copyright" and "Description" are all self-explanatory. There should, finally, be an input field for each of your inputs (currently only one is supported) and an output field for each output. What comes in and what goes out of these ports should also be briefly described.


Now, with these resources in your projects resource file, you are ready to compile your filter. Compile it as a "Code Resource" with a resource type of 'FILF' and ID of 128. Assign the resulting resource file a type code of 'FILF' and creator code of 'FTOP'. Depending on your code and your development environment, you may have to include one or more libraries in your project. Check the documentation if necessary.

That's it! You now have a FilterTop filter that will be fully recognized by FilterTop, can be added to any pipeline and will work harmoniously with all of the existing filters. Just drag it into the appropriate subfolder in the "Filters" folder. Again, all of the above steps take longer to describe than to do. With practice, all of this will become second nature as you build up a collection of useful filters, each easy to write because FilterTop handles all of the drudgery for you. Now you can devote all of your creativity to that translation or encryption utility (or whatever) that you've been hankering to write. Feel free to submit your work to TopSoft. It may just be added (with your permission of course) to the growing collection of free filters available at the TopSoft FTP site.

The FilterTop area of the TopSoft FTP site is located at <http://ftp.topsoft.org/Visitors/FilterTop/>. There you will find the complete FilterTop application (it's free!) as well as various filters not included in the main package. Also of special interest are the archive of filter source code (see what others have done) and the Filter Writer's Guide, which provides much more information about writing filters than I have been able to give here, including guidelines for creating your own custom icons for your filters and for writing filters that have more specialized requirements than our simple example. You can also easily create help documents that are modular like filters and are automatically added to FilterTop's elaborate About box. Also check out the Web site at <http://www.topsoft.org>. Information about anything you can't find at the FTP or Web site may be obtained directly from Tony Jacobs, the President of TopSoft, at [info@topsoft.org](mailto:info@topsoft.org).

Join the plug-in revolution now!

### ACKNOWLEDGMENTS

Thanks to John Tsombakos for his helpful feedback, which resulted in a number of improvements in this article; to Tony Jacobs for his encouragement and for urging me to write the article in the first place; and to all members of the FilterTop development team, past and present, without whose phenomenal efforts none of this would have been possible. 





by Tim Monroe and Bryce Wolfson, Apple Computer, Inc.

# Programming With QuickTime VR

## *A look at the new API for managing QuickTime VR movies*

### INTRODUCTION

QuickTime VR is the part of the QuickTime Media Layer that allows users to interactively explore and examine photorealistic, three-dimensional virtual worlds and objects. For several years, MacOS and Windows users have been able to play back QuickTime VR content using standard navigation controls. With version 2.0, QuickTime VR now supports a C API that allows developers to customize and extend the user's virtual experience. Here you'll find everything you need to know to get started supporting QuickTime VR in your application.

QuickTime VR is certainly one of the hottest Apple technologies today. Dozens and dozens of CD titles have appeared in the past months that depend on QuickTime VR. More significantly, the World Wide Web now serves up literally thousands of QuickTime VR movies. A large part of this popularity stems from the fact that the amount of data required to display a complex, photorealistic panorama using QuickTime VR is much smaller than would be required to model that panorama in detail using a standard 3D graphics

application. This, together with the fact that no run-time rendering is occurring, drastically reduces the amount of CPU horsepower and RAM required to immerse the user in a realistic 3D environment. QuickTime VR can transport the user from the tightest passageway deep inside the pyramids to the wide-open expanse of the space shuttle payload bay.

But, to paraphrase Al Jolson, you ain't seen nothin' yet, folks. Apple has recently introduced a C language programming interface to QuickTime VR — called the QuickTime VR Manager — that provides a large set of tools for extending and customizing the user's virtual experience. The QuickTime VR Manager provides the necessary hooks for you to add your own custom processing to VR movie playback and to integrate QuickTime VR content playback with other technologies, particularly with other multimedia technologies. For instance, you can very easily integrate Apple's SoundSprocket with QuickTime VR to attach sounds to specific locations in a panorama. Or, you can play QuickTime movies on the screen of a television in a panorama. Or, you can embed QuickDraw 3D objects (even moving objects!) in a panorama or object node. Take your favorite technology and, chances are, there's a cool way to integrate it with a VR panorama or object.

In this article, we'll explain some basic ways of using the QuickTime VR Manager in your application. We start by reviewing some simple but important aspects of QuickTime movie support. Then we move into programmatic control of QuickTime VR movies by showing how to use standard Movie Toolbox and movie controller functions to operate on QuickTime VR movies. Finally, we'll describe the QuickTime VR Manager itself and present some code that demonstrates a few of its capabilities.

Before reading this article, you should be familiar with the basic capabilities and operations of QuickTime VR. At the very least, you should have opened some panoramas or object

---

**Tim Monroe** <monroe@apple.com> is a software engineer on the QuickTime VR team, responsible for developing sample code for the new QuickTime VR C language API. In his previous life at Apple, he worked on the Inside Macintosh team.

**Bryce Wolfson** <bwolfson@apple.com> is also a software engineer with Apple's QuickTime VR team. He's responsible for parts of the QTVR runtime's architecture, human interface, and application interaction, and has been known to occasionally write bits of over-commented sample code.



movies using a QuickTime player (for instance, MoviePlayer or QTVRPlayer) and have a good understanding of the VR user interface (panning and tilting, zooming in and out, triggering hot spots, and so forth). You can also get the necessary background information by looking at the first dozen pages of the book *Virtual Reality Programming With QuickTime VR 2.0*, the developer documentation for the QuickTime VR Manager.

### BACK TO BASICS

The first thing to keep in mind is that QuickTime VR movies are just QuickTime movies, at least in the way the movie data is stored in files. The image data for panoramas and objects is stored in standard QuickTime video tracks in movie files. Other data (such as names of the nodes in a scene) is stored in data atoms in the movie resources. What's different about QuickTime VR is the way in which the image data is handled. In a QuickTime movie, frames of a movie are read from a video track and displayed in sequence to the user. In a QuickTime VR movie, the image handling is more complicated, because the image to be displayed depends more heavily on user interaction. In an object movie, for instance, the user can pan left or right, tilt up or down, or zoom in and out (among other things). These actions typically require a new frame from the movie's video track to be displayed. The new frame, however, may or may not follow the current frame in the movie's video track.

Displaying the appropriate panorama or object images in response to user actions is the responsibility of the QuickTime VR movie controller, which is a standard QuickTime movie controller. Every QuickTime VR movie contains a special piece of user data that identifies the movie controller for the movie. This user data is examined by the Movie Toolbox when an application calls the `NewMovieController` function to create a movie controller for the movie. If the QuickTime VR extension is installed and the QuickTime VR movie controller component is therefore available, the Movie Toolbox locates that controller and assigns it to that movie.

Now here's the payoff: if you've done things right, *your QuickTime application already supports QuickTime VR!* Go ahead and try it: launch your QuickTime-savvy application, select Open from the File menu, and then choose a QuickTime VR movie. (Easier still, just drag the VR movie icon onto your application's icon.) If everything goes according to plan, the VR movie will open correctly and you'll be able to navigate within the panorama or manipulate the object in all the right ways. This is because, as we've just said, the QuickTime VR file contains information that specifies the QuickTime VR movie controller instead of a QuickTime movie controller. It couldn't be any easier.

This payoff, however, has a catch: to get VR movie playback, you have to support QuickTime in the right way. Part of what that means is that you have to call `NewMovieController` to associate a movie controller with a movie. Years ago, Peter Hoddie voiced this warning in a *develop* column: "When you need a user interface for playing a movie, you should use `NewMovieController` to create a movie controller appropriate for use with that movie. A common mistake is to instead use the Component Manager routine `FindNextComponent` or `OpenDefaultComponent` to locate a movie controller. This finds the first movie controller in the system's list

of registered components. QuickTime has always contained only one movie controller, so this worked fine. However, future versions of QuickTime will almost certainly include other movie controllers, so the first one isn't necessarily the most appropriate one." (Hoddie, p. 22)

These "future versions" of QuickTime are here now, and one of them is QuickTime VR. So, to support VR movies, you must call `NewMovieController` instead of the Component Manager. More generally, you should make sure that you've followed all the advice in Hoddie's important article. It won't take you long to do so, and it will make it easier to support QuickTime in all its manifestations.

### BEYOND THE BASICS

Once you've implemented basic — and correct — QuickTime movie playback support in your application, there are still a few other things you should do to handle QuickTime VR movies. In particular, you'll want to enable and disable menu items correctly and to display the correct cursor when it's outside the movie window. Happily, you can do each of these things with just a few lines of code. We're supposing throughout that you want your application to handle *both* QuickTime and QuickTime VR movies; if instead you wanted to build just a QuickTime VR player, some of these steps (for instance, worrying about the Select All menu command) wouldn't be necessary.

The first thing you'll want to do is support the standard movie editing commands for QuickTime movies but also disable any Edit menu items that don't make sense for QuickTime VR movies. When you first open a QuickTime or QuickTime VR movie, you can call `MCEnableEditing` to turn on editing (which by default is off):

```
MCEnableEditing(myMC, true); // enable movie controller editing
```

Here `myMC` is the movie controller. We'll keep track of this and other information by setting fields in a window object record, a data structure associated with each window that contains a QuickTime or QuickTime VR movie; see later on in this article for the exact structure of a window object record.

As recommended in Hoddie's article, we use the `MCSetUpEditMenu` function in our menu-adjusting routine to enable and disable items in the Edit menu:

```
myMC = (**myWindowObject).fController;
if (myMC != NULL)
    MCSetUpEditMenu(myMC, 0L, GetMHandle(mEdit));
```

The QuickTime VR movie controller disables most of the Edit menu items, since they don't apply to VR movies. It does not however disable the Select All item, so you must do that yourself, like this:

```
myErr = MCGetControllerInfo(myMC, &myControllerFlags);
if (myErr != noErr)
    myControllerFlags = 0L;
isEditingEnabled =
    (mcInfoEditingEnabled & myControllerFlags) != 0;
if (isEditingEnabled)
    EnableItem(GetMHandle(mEdit), iSelectAll);
else
    DisableItem(GetMHandle(mEdit), iSelectAll);
```



Next, the QuickTime VR movie controller manages the cursor whenever it's within a VR movie, changing its shape at appropriate times (such as when the cursor is over a hot spot). The QuickTime VR movie controller doesn't restore the cursor to the arrow shape when the cursor moves outside of the movie. To do that, you can put this code in your idle event processing code:

```
if (theWindow == FrontWindow()) {
    Rect myRect;
    GetMouse(&myPoint);
    MCGetControllerBoundsRect(myMC, &myRect);
    if (!PtInRect(myPoint, myRect))
        InitCursor();
}
```

This cursor-adjusting code is currently needed only for QuickTime VR movies, but should probably be called by any QuickTime-savvy application.

You'll notice that we've used several movie controller component functions with a QuickTime VR movie. This underscores once again the fact that QuickTime VR movies are just special kinds of QuickTime movies. For instance, you can hide the controller bar by executing this code:

```
MCS setVisible(myMC, false);
```

In general, you can use any movie controller functions with QuickTime VR movies, except those that specifically relate to time or movie editing. It would make no sense, for example, to issue the movie controller action `mcActionGoToTime` on a QuickTime VR movie. When QuickTime VR movies do involve temporal aspects (for instance, an animated object movie has a play rate), the QuickTime VR Manager provides functions that you can use to manipulate those aspects.

The QuickTime VR movie controller also supports movie controller action filters. That is, you can intercept a VR movie's controller actions and react accordingly. So virtually the entire QuickTime programming interfaces are applicable also to QuickTime VR movies.

So far, we've shown how to support QuickTime VR movie playback, and even provide a significant level of customization of the movie playback and user interface, just using standard QuickTime movie controller functions. The sample application `VRShell` provides a concrete example of this capability. It's built on `MovieShell`, a QuickTime-savvy framework written by Apple DTS. All of our remaining sample applications are built in turn on `VRShell`.

#### OVERVIEW OF THE QUICKTIME VR MANAGER

The QuickTime VR Manager provides a large number of capabilities that you can use to customize and extend the user's virtual experience of panoramas and objects. Here we'll summarize the basic capabilities of the QuickTime VR Manager. Then, in the following sections, we'll illustrate how to use some of them. In this article, we'll keep things fairly simple; in the future, we hope to illustrate some of the more advanced capabilities of the QuickTime VR Manager.

# Name Your Poison

Java OpenDoc  
(ODF)  
MPW PowerPlant  
Roaster  
codeWarrior  
SYMANTEC  
neoAccess  
TCL  
OOFILE  
Tools Plus

**AppMaker** supports most popular languages and frameworks. Just point and click to design your user interface, then let AppMaker create resources and generate "human, professional quality code" to implement your design.

Use AppMaker as a prototyping and productivity tool or use it as a learning tool for Mac programming techniques or for new environments such as OpenDoc, Java, or PowerPlant.

AppMaker is just \$299 and includes a one-year subscription on CD.

**B • O • W • E • R • S  
Development**

**603-863-0945**

[bowersdev@aol.com](mailto:bowersdev@aol.com)

<http://members.aol.com/bowersdev>



The QuickTime VR Manager provides these main capabilities:

- **Positioning.** A VR movie file contains a scene, which is a collection of one or more nodes. Each node is either a panoramic node (a "panorama") or an object node (an "object") and is uniquely identified by its node ID. Within a panoramic node, the user's view is determined by three factors: the pan angle, the tilt angle, and the vertical field of view. For objects, the view is also determined by the view center. The QuickTime VR Manager provides functions to get and set any of these items. For instance, you can programmatically spin an object around by repeatedly incrementing the current pan angle.
- **Hot spot handling.** You can use the QuickTime VR Manager to manage any hot spots in a panorama or object. For instance, you can trigger a hot spot programmatically (that is, simulate a click on the hot spot), enable and disable hot spots, determine whether the cursor is over a hot spot, find all visible hot spots, and so forth. You can also install a callback routine that is called whenever the cursor is over an enabled hot spot.
- **Custom node entering and leaving behaviors.** The QuickTime VR Manager allows you to perform actions whenever the user enters a new node or leaves the current node. For instance, you might play a custom sound when the user enters a particular node. Current versions of QuickTime VR support scenes that contain both panoramic and object nodes. If you want to treat object nodes differently from panoramic nodes, you can use node-entering and node-leaving procedures to do any necessary processing.
- **Getting information.** You can use the QuickTime VR Manager to get information about a scene or about a specific node. For instance, you might want to determine the ID and type of the current node. Much of the information about scenes and nodes is stored in atoms in the movie file. To get information about a scene or node that isn't provided directly by the QuickTime VR Manager, you'll have to use the QuickTime atom container functions (introduced in QuickTime version 2.1) to extract information from those atoms.
- **Intercepting QuickTime VR Manager functions.** You can intercept calls to some QuickTime VR Manager functions to augment or modify their behavior. For example, to assign behaviors to custom hot spots, you can install an intercept routine to be called whenever a hot spot is triggered. Your intercept routine would check the type of the triggered hot spot and then perform the correct actions for that type. Another common use of intercept routines is to intercept positioning functions (changing the pan, tilt, and field of view) and adjust environmental factors accordingly.
- **Accessing offscreen panorama buffers.** QuickTime VR maintains two offscreen buffers for each panorama, a back buffer and a prescreen buffer. The back buffer contains some or all of the data in a panorama image track, which is a standard QuickTime video track whose frames contain slices of a cylindrically warped and rotated version of the original panorama. The prescreen buffer contains the unwarped and unrotated image that is about to be copied to the screen.

You can use QuickTime VR Manager functions to access the back buffer and the prescreen buffer. Which buffer you draw into is determined by the effect you want to achieve. To overlay a graphic (such as a head's-up display) that is unaffected by the user's panning, tilting, or zooming, you would draw into the prescreen buffer. To overlay a graphic that is affected by changes in the view, you would draw into the back buffer. Back buffer drawing is a bit tricky, however, because the images you draw there must be rotated and warped if they are to appear correctly on the screen. (Note: in future versions of QuickTime VR, the back buffer might not be rotated. The QuickTime VR Manager provides a way to check whether it is rotated or not.)

This list is not exhaustive. The QuickTime VR Manager provides many other capabilities as well. For a complete description, see the book *Virtual Reality Programming With QuickTime VR 2.0*.

## STARTING UP

Before you can do these neat things with the QuickTime VR Manager, you must do a little setting up (over and beyond what's required for using QuickTime). First, of course, you must ensure that the QuickTime VR Manager is available in the current operating environment. As you'd expect, there are several Gestalt selectors that you can use to see whether the QuickTime VR Manager is available and what features it has. Here we'll spare you the details on calling Gestalt; consult the sample code (or the reference book) if you absolutely can't live without seeing them.

The QuickTime VR Manager keeps track of QuickTime VR movies using an identifier called a QTVR instance (of data type `QTVRInstance`). Virtually all QuickTime VR Manager calls operate on QTVR instances. You can think of an instance as representing a scene — that is, a collection of nodes, or sometimes just the current node. You get a QTVR instance by calling the `QTVRGetQTVRInstance` function, as shown in Listing 1. `QTVRGetQTVRInstance` takes a reference to a QTVR track, which you can get by calling `QTVRGetQTVRTrack`. (In general, you'll never do anything else to the QTVR track, so you can safely forget about it.)

### Listing 1: `InitApplicationWindowObject`

```
void InitApplicationWindowObject
(WindowObject theWindowObject)
{
    Track          myQTVRTrack = NULL;
    Movie          myMovie = NULL;
    MovieController myMC = NULL;
    QTVRInstance    myInstance = NULL;

    if (theWindowObject == NULL)
        return;
    // make sure we can safely call the QTVR API
    if (!gQTVRMgrIsPresent)
        return;
    // find the QTVR track
    myMovie = (**theWindowObject).fMovie;
    myMC = (**theWindowObject).fController;
    myQTVRTrack = QTVRGetQTVRTrack(myMovie, 1);
    // get a QTVR instance and remember it
    QTVRGetQTVRInstance(&myInstance, myQTVRTrack, myMC);
    (**theWindowObject).fInstance = myInstance;
    // do any QTVR window configuration
    if (myInstance != NULL) {
        // set units to radians
        QTVRSetAngularUnits(myInstance, kQTVRRadians);
    }
}
```



The `InitApplicationWindowObject` function defined in Listing 1 takes as a parameter a window object, which is a handle to a data structure associated with each window containing a QuickTime (or QuickTime VR) movie:

```
typedef struct {
    WindowRef      fWindow;          // the window
    OSType          fObjectType;      // app-specific window tag
    Movie           fMovie;           // the main movie (QT or QTVR)
    MovieController fController;      // the movie controller
    FSSpec          fFileFSSpec;      // the file's FSSpec
    short           fFileResID;       // the file's resource ID
    short           fFileRefNum;       // the file's reference number
    QTVRInstance    fInstance;        // the QTVRInstance
    Handle          fAppData;         // a handle to app-specific data
} WindowObjectRecord, *WindowObjectPtr, **WindowObject;
```

The fields of this structure contain, among other things, references to the movie and movie controller, and the QTVR instance. The field `fAppData` is a handle to any other data that might have to be associated with the window. For the time being, we'll ignore that field.

Notice that Listing 1 calls the `QTVRSetAngularUnits` function. The QuickTime VR Manager can work with either degrees or radians when specifying angular measurements. The default angular unit type is degrees, but you can change the current unit type by calling `QTVRSetAngularUnits`. Internally, the QuickTime VR Manager always uses radians, and in some situations gives you measurements in radians no matter what the current angular unit. In general, therefore, we find it easier to work in radians most of the time, so we reset the angular unit type to radians. (Your preference may vary!) Of course, we can define some macros to convert degrees to radians and vice versa, should the need arise:

```
#define kVRPi          ((float)3.1415926535898)
#define kVR2Pi        ((float)(2.0 * 3.1415926535898))
#define QTVRUtils_DegreesToRadians(x) \
    ((float)((x) * kVRPi / 180.0))
#define QTVRUtils_RadiansToDegrees(x) \
    ((float)((x) * 180.0 / kVRPi))
```

### WHERE AM I?

Finally, we're ready to use the QuickTime VR Manager to do some real work. The most basic way to use the API is to control the view parameters of a node — the pan, tilt, and field of view angles. Listing 2 defines a function that gradually increments the pan angle through 360 degrees. With panoramas, this has the effect of making the user seem to spin a full circle (as if the user is spinning on a rotating stool). With objects, this has the effect of making the object spin around a full circle (as if the object is spinning on a lazy Susan).

#### Listing 2: SpinAroundOnce

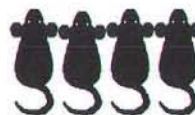
```
void SpinAroundOnce (QTVRInstance theInstance)
{
    float    myOrigPanAngle, myCurrPanAngle;

    myOrigPanAngle = QTVRGetPanAngle(theInstance);
    for (myCurrPanAngle = myOrigPanAngle;
        myCurrPanAngle <= myOrigPanAngle + kVR2Pi;
        myCurrPanAngle += QTVRUtils_DegreesToRadians(10.0)) {
        QTVRSetPanAngle(theInstance, myCurrPanAngle);
        QTVRUpdate(theInstance, kQTVRCurrentMode);
    }
}
```

The professional web page development tool

- Award winning spell checker
- True, continuously updated, WYSIWYG preview
- Tables (creating, editing, and importing)
- Limited site management
- Forms
- Page colors
- Frames
- Text and styled text importing
- Add your own tags
- Very customizable
- Preview with different browsers
- Includes two free upgrades

## World Wide Web Weaver



MacUser, August 1996

## SiteWeaver

- Check/Edit/Update all of your links on your site on seconds!
- Upload pages and sites directly to the web
- Work offline and then let SiteWeaver™ synchronize your work later
- Easy Mac interface for novices or full time professionals
- Works with just about any graphics/plugin program, page editor\* and web browser

For the latest ordering information point your web browser to <http://www.MiracleInc.com> or call (315) 265-0930.

The idea here is simple: get the starting pan angle (by calling `QTVRGetPanAngle`) and then repeatedly increment the pan angle by a certain amount (here, 10 degrees) until a full circle has been traversed. Note that we must call the `QTVRUpdate` function after we set a new pan angle to make sure the updated view is displayed on the screen.

### OVERLAYING IMAGES ON A PANORAMA

Suppose you wanted to display a logo or other graphical element in the corner of a QuickTime VR panoramic movie (in the way that some TV channels often do to discourage pirating). Because the overlaid logo isn't affected by the view settings, we can just draw it into the panorama's prescreen buffer.

We must keep track of the picture to be drawn, so we'll create an application data record and store a handle to that record in the `fAppData` field of the window object record. For present purposes, our data record can look like this:

```
typedef struct {
    PicHandle fPicture;          // the picture to display
    Boolean fDisplayPicture;     // is the picture to be displayed?
} ApplicationDataRecord, *ApplicationDataPtr,
**ApplicationDataHdl;
```

We'll get the overlay picture from a PICT resource, in a window data initialization routine (shown in Listing 3) called by the application's `InitApplicationWindowObject` function.



# WHY INSTALLER VISE?

## ASK METROWERKS.

*"Installing CodeWarrior is a herculean task, we ship over 400 megabytes on each CodeWarrior CD release. It would frankly be impossible for us to undertake this task three times a year without Installer VISE from MindVision. VISE is simply a godsend."*

—Greg Galanos, President, Metrowerks Corporation

## Installer VISE.

### The Professionals' Choice.



[www.mindvision.com](http://www.mindvision.com)



#### Listing 3: VRLogo\_InitWindowData

```
ApplicationDataHdl VRLogo_InitWindowData
(WindowObject theWindowObject)
{
    ApplicationDataHdl    myAppData;
    myAppData = (ApplicationDataHdl)
        NewHandleClear(sizeof(myAppData));
    if (myAppData != NULL) {
        // get the picture to overlay
        (**myAppData).fPicture = GetPicture(kPictureResID);

        // set initial display state
        (**myAppData).fDisplayPicture = true;
    }
    return(myAppData);
}
```

Also, in the application's InitApplicationWindowObject function, we must install our prescreen buffer imaging completion procedure, which is called by the QuickTime VR Manager each time the prescreen buffer is about to be copied to the screen. We can install our procedure like this:

```
if (QTVRGetNodeType(myInstance, kQTVRCurrentNode)
    == kQTVRPanoramaType)
{
    ImagingCompleteUPP myImagingProc;
    myImagingProc =
        NewImagingCompleteProc(VRLogo_PrescreenRoutine);
    QTVRSetPrescreenImagingCompleteProc(myInstance,
        myImagingProc, (SInt32)theWindowObject, 0);
}
```

The third parameter to QTVRSetPrescreenImagingCompleteProc is an application-specific reference constant value; here we pass the window object reference, so the prescreen buffer can access the data associated with the window.

Our prescreen buffer imaging completion procedure is called after QuickTime VR has finished drawing into the prescreen buffer. When it's called, the current graphics port is set to the prescreen buffer. All we need to do is draw the picture at the appropriate spot, as shown in Listing 4.

#### Listing 4: VRLogo\_PrescreenRoutine

```
pascal OSErr VRLogo_PrescreenRoutine
(QTVRInstance theInstance, WindowObject theWindowObject)
{
    #pragma unused(theInstance)

    ApplicationDataHdl    myAppData;
    Rect                  myMovieRect;
    Rect                  myPictRect;

    // get the application-specific data associated with the window
    myAppData = (ApplicationDataHdl)
        GetAppDataFromWindowObject(theWindowObject);
    if (myAppData == NULL)
        return(paramErr);

    // if there is no picture to display or displaying is toggled off, just return
    if ((**myAppData).fPicture == NULL)
        return(noErr);

    if (!(**myAppData).fDisplayPicture)
        return(noErr);

    // get the current size of the movie
    GetMovieBox((**theWindowObject).fMovie, &myMovieRect);

    // set the size and position of the overlay rectangle
    SetRect(&myPictRect, 0, 0, 32, 32);
    OffsetRect(&myPictRect,
        myMovieRect.right - (myPictRect.right + 5),
        myMovieRect.bottom - (myPictRect.bottom + 5));

    // draw the picture
    DrawPicture((**myAppData).fPicture, &myPictRect);

    return(noErr);
}
```

There's nothing very complicated in this prescreen buffer imaging completion procedure. Essentially, it just figures out where in the buffer to draw the picture and then draws it.

Note that the current release of the QuickTime VR Manager maintains prescreen buffers only for panoramic nodes. It's possible, however, with a little effort, to create your own prescreen buffer for object nodes and then perform the same kind of overlays that are possible with panoramic nodes.

#### INTEGRATING WITH OTHER MEDIA

Much of the real power provided by the QuickTime VR Manager derives from the ease with which it allows you to integrate VR movies with other media, such as video, sound, and 3D objects. In a future article, we'll show how to embed QuickDraw 3D objects in a panorama. In the meantime, we'll give you a good taste of what's possible by showing how to integrate QuickTime VR and SoundSprocket, the part of Apple Game Sprockets that supports localized sounds (that is, sounds emanating from a specific location in a panorama). We'll suppose



that you're already familiar with SoundSprocket, but the ideas are so simple that you can probably follow along even if you aren't.

SoundSprocket provides a virtual audio environment consisting of a single listener and one or more sound sources. The listener and the sound sources all have independent positions in 3D space. In addition, the listener and sound sources all have independent orientations in 3D space. The basic idea behind our sample application is that the listener is situated at the nodal point of the panorama (the point around which the panorama turns) and is looking at the center of the movie window. As the user interactively changes the pan and tilt angles of the panorama, the fixed locations of the sound sources change relative to the listener. (SoundSprocket doesn't actually require that the sound sources have fixed locations, but we'll keep the locations of our sources fixed, for simplicity.)

The data we need to maintain for each VR movie has this structure:

```
typedef struct {
    SSplListenerReference    fListener;
    SSplSourceReference      fSources    [kMaxNumSourcesPerNode];
    SndChannelPtr            fChannels   [kMaxNumSourcesPerNode];
    SndListHandle            fResources  [kMaxNumSourcesPerNode];
    float                    fPrevPanAngle;
    float                    fPrevTiltAngle;
    ApplicationDataRecord,   *ApplicationDataPtr,
    **ApplicationDataHdl;
```

We're keeping track of the listener and the sound sources used by SoundSprocket, as well as a sound channel and a sound resource for each sound in the panorama. Finally, we're keeping track of the previous pan and tilt angles, which we'll compare with the current pan and tilt angles to determine whether we need to update the listener's orientation.

We'll skip over the details of setting up the virtual audio environment. What's of interest here is the way in which we translate changes in the VR movie's pan and tilt angles into changes in the listener's orientation. We do this using a prescreen buffer imaging completion procedure, not because we want to actually draw anything into the prescreen buffer, but simply because we want to be called whenever the pan or tilt angles of a panorama have changed and (therefore) a new view is about to be displayed. Our prescreen routine is shown in Listing 5.

#### Listing 5: VR3DSound\_PrescreenRoutine

```
pascal OSErr VR3DSound_PrescreenRoutine
(QTVRInstance theInstance, WindowObject theWindowObject)
{
    float                myPan;
    float                myTilt;
    TQ3Vector3D          myOrientation;
    ApplicationDataHdl    myAppData;

    myAppData = (ApplicationDataHdl)
        GetAppDataFromWindowObject(theWindowObject);
    if (myAppData == NULL)
        return(paramErr);

    // get the current pan and tilt angles (in radians)
    myPan = QTVRGetPanAngle(theInstance);
    myTilt = QTVRGetTiltAngle(theInstance);
```

```
// determine whether the pan or tilt angle has changed
if ((myPan == (*myAppData).fPrevPanAngle) &&
    (myTilt == (*myAppData).fPrevTiltAngle))
    return(noErr);

(*myAppData).fPrevPanAngle = myPan;
(*myAppData).fPrevTiltAngle = myTilt;

// figure out the new orientation of the listener
myOrientation.x = -sin(myPan) * cos(myTilt);
myOrientation.y = sin(myTilt);
myOrientation.z = cos(myPan) * cos(myTilt);

// set the new orientation of the listener
SSplListener_SetOrientation
    ((*myAppData).fListener, &myOrientation);

// update the virtual audio environment
VR3DSound_Update3DSoundEnv(theWindowObject);

return(noErr);
}
```

Once again, there isn't anything very complicated here. This prescreen routine gets the current pan and tilt angles and then (using a little elementary trigonometry) converts those angles into a point in three-dimensional space. For simplicity, we've assumed that the sound sources are all located one unit away from the listener, but it would be quite trivial to remove that restriction. Then the prescreen routine sets the new orientation for the listener and updates the virtual audio environment.

The source code for the VR3DSound sample application contains (in addition to all the necessary SoundSprocket processing) code for opening sound resources and stopping a node's sounds when the user moves to a new node. That code illustrates how to use node-entering and node-leaving procedures.

### INTERCEPTING QUICKTIME VR MANAGER FUNCTIONS

Suppose you want to play a sound every time the user clicks (that is, triggers) a hot spot. The easiest way to do this is to install an intercept procedure that is called each time a hot spot is triggered. The intercept procedure simply plays the sound and then returns, whereupon QuickTime VR processes the hot spot click as usual. Listing 6 shows a simple hot spot triggering intercept procedure.

#### Listing 6: VRSample\_InterceptRoutine

```
pascal void VRSample_InterceptRoutine (
    QTVRInstance theInstance,
    QTVRInterceptPtr theMsg,
    WindowObject theWindowObject,
    Boolean *cancel)
{
    #pragma unused(theInstance, theWindowObject)

    Boolean    myCancelInterceptedProc = false;
    switch (theMsg->selector) {
        case kQTVRTriggerHotSpotSelector:
            MyPlaySound();
            break;

        default:
            break;
    }

    *cancel = myCancelInterceptedProc;
}
```



# TestTrack

Your Total Bug Tracking Solution

## Introducing TestTrack 1.5 and Solo Bug!

Discover the tool more and more of today's top software developers are using to improve the quality of their Mac and Windows applications—**TestTrack**.

- Track bugs, feature requests, customer information, and more.
- Use on Mac or Windows 95/NT platforms.
- Produce concise reports.
- Link engineers, testers, managers, even tech writers.
- Automatically route bugs to team members.
- Save time and improve tech support by giving **Solo Bug**, our stand-alone bug reporter, to your customers (comes free with TestTrack 1.5).
- Track the history of each bug.
- Multiple users, full security!
- Ideal for consultants too!

To order call 513-683-6456

Seapine Software, Inc. 1066 Seapine Ct. Maineville, OH 45039

sales@seapine.com  
http://www.seapine.com

 Seapine  
Software

Powerful enough to  
handle all your bug  
tracking needs.  
Affordable enough to  
put on everyone's  
desktop.

An intercept routine is executed whenever the intercepted routine is called, either programmatically or by a user action. (We'll show you shortly how to specify which routine or routines you want to intercept.) On entry, the QuickTime VR Manager provides three pieces of information: the relevant QTVR instance, a pointer to an intercept record, and an application-defined reference constant, which we use here to pass in the window object. The intercept record (pointed to by the `theMsg` parameter) has this structure:

```
typedef struct QTVRInterceptRecord {  
    SInt32 reserved1;  
    SInt32 selector;  
    SInt32 reserved2;  
    SInt32 reserved3;  
    SInt32 paramCount;  
    void *parameter[6];  
} QTVRInterceptRecord; *QTVRInterceptPtr;
```

For present purposes, we need inspect only the selector field, which contains a constant that indicates which intercepted routine is being called. As you can see in Listing 6, we simply look for any calls to `QTVRTriggerHotSpot` and call the application-defined function `MyPlaySound` when we get one.

You can install an intercept procedure by calling the `QTVRInstallInterceptProc` function, as shown in Listing 7.

### Listing 7: VRSample\_InstallInterceptRoutine

```
void VRSample_InstallInterceptRoutine (  
    QTVRInstance theInstance,  
    WindowObject theWindowObject)  
{  
    QTVRInterceptUPP myInterceptProc;  
  
    myInterceptProc =  
        NewQTVRInterceptProc(VRSample_InterceptRoutine);  
  
    QTVRInstallInterceptProc(theInstance,  
        kQTVRTriggerHotSpotSelector,  
        myInterceptProc,  
        (SInt32)theWindowObject, 0);  
}
```


### VIRTUALLY FINISHED

It seems like we've barely scratched the surface of the QuickTime VR Manager, but even so we've illustrated some very powerful capabilities for managing QuickTime VR movies programmatically. We've shown how to perform basic positioning of the viewer, how to alter the displayed image by drawing into a panorama's prescreen buffer, how to link the orientation of a listener in SoundSprocket's virtual audio environment to the QuickTime VR view angles, and how to intercept some QuickTime VR Manager functions. Not bad for just over a hundred lines of code!

We mentioned at the outset that the QuickTime VR Manager allows you integrate QuickTime movies and QuickDraw 3D objects with QuickTime VR panoramas and objects. Now that you've seen how to use the VR API, and particularly how to support SoundSprocket, you can probably figure out how to do at least the QuickDraw 3D integration yourself. If not, don't despair. Just have some patience until we show you — in our next article — how to play movies and render 3D objects in a QuickTime VR panorama.

### BIBLIOGRAPHY AND REFERENCES

Apple Computer, Inc. *Virtual Reality Programming With QuickTime VR 2.0* (1997). Cupertino, CA.

Hoddie, Peter. "Somewhere in QuickTime: Basic Movie Playback Support". develop, *The Apple Technical Journal*, issue 18 (June 1994), pp. 22–25. Apple Computer's Developer Press. 

Want to suggest an article for the  
magazine? Send your suggestion to  
<mailto:editorial@mactech.com>





by Tim Monroe, Apple Computer, Inc.

# The QuickTime Media Layer

## *An overview of Apple's flagship multimedia software*

### INTRODUCTION

Imagine that you're standing in a museum, surrounded by famous works of art. On the wall in front of you hangs a painting by Jackson Pollack. You reach out and touch the painting, and instantly a screen slides down in front of the painting; on the screen appears a video showing Pollack in the process of splashing paint and stabbing wildly at the canvas. Soon it becomes clear that he's working on the very painting that hangs in front of you.

You move to the next room. As you enter, a voice announces that this room contains the last few remaining vases from the Ming dynasty. You go to the center of the room and walk around the blue and white vase that sits on a pedestal. Then you *pick up* the vase to examine it further: you want to see the bottom and to look inside the vase. Suddenly, off to the right and a bit behind you, you hear a crash as a vase collides with the marble floor. As you turn and look, a crowd of schoolchildren hurriedly moves away from a pile of shards. Unfazed, you utter the phrase "reassemble the vase, please." The shards swirl up into a spinning cloud that eventually settles gracefully onto its pedestal as a complete, unbroken vase.

Then you move into a third room. Large mobiles hang from the ceiling. You click a switch on the wall and a fan begins to spin, gently blowing the mobiles and setting them in motion. The fan hums and the mobiles creak and clang as they move and collide with one another. After a few minutes, the fan stops abruptly. A wall, previously blank, now displays a live video picture of a security guard announcing that the museum will be closing soon. You leave the museum.

This experience — this *multimedia* experience — seemed so real that for a few moments you perhaps forgot that you were sitting in front of a computer. You almost felt sorry for the schoolchildren when they accidentally smashed the vase. And you almost even thought you could feel that hard marble floor under your feet as you moved around in your virtual museum. What made this experience real, what made it seem as if you were immersed in another world, was the set of Apple multimedia technologies collectively known as the QuickTime Media Layer (QTML). In this article, I'll provide an overview of the QTML. I'll describe the main components of the QTML and highlight some of the key interrelations among those components.

### WHAT IS QTML?

In a nutshell, the QuickTime Media Layer is a collection of technologies developed by Apple Computer that allow authoring and playback of multimedia content. There are three main technologies currently included in the QTML: QuickTime, QuickDraw 3D, and QuickTime VR. (As we'll see later, there are several other technologies that are loosely associated with the QTML.) What distinguishes each of these technologies, and makes them suitable for inclusion in the QTML, are these features:

- The technology is *media-rich*. The QTML provides, first and foremost, an avenue for the delivery of digital media. QTML technologies pertain to the eyes and ears. Other sensory modes, if digitized, would be prime candidates for inclusion in the QTML. For instance, some joysticks can provide tactile

**Tim Monroe** <monroe@apple.com> is a software engineer on Apple's QuickTime VR team, responsible for developing sample code for the new QuickTime VR C language API. In his previous life at Apple, he worked on the Inside Macintosh team, where he wrote developer documentation for QuickDraw 3D, QuickTime VR, the sound and speech technologies, and a host of other APIs.



feedback, and a general interface to such devices would make a natural QTML component. We'll probably have to wait a while, however, for olfactory (QuickSmell?) and gustatory components (QuickTaste?) of the QTML.

- The technology is *interactive*. It's great to sit and watch things happen, but it's essential for most everyday uses of media delivery that the user be able to interact with the environment. Accordingly, the core QTML technologies provide some means for users to control the environment. QuickDraw 3D provides a picking architecture that allows the user to move or select objects in a scene. QuickTime VR is based almost entirely on allowing the user to choose a navigation path through a virtual world or to manipulate an object in a virtual world. Even QuickTime — long the prime example of “just sit and watch” — is moving toward increased interactivity.
- The technology is *cross-platform*. The QTML is based on a strategy of “author once, deliver many.” Ideally, applications developers should be able to build playback applications for the major personal computer operating systems, including MacOS, Windows 95, and Windows NT. (QuickTime itself currently also runs on OS/2 and several flavors of UNIX.) Just as importantly, developers should be able to build authoring systems for any of these platforms. Both of these needs can be served by providing a set of cross-platform application programming interfaces (APIs) for QTML components.
- The technology supports a standard *file format*. This feature is just as important as the cross-platform APIs just mentioned and provides a key part of the QTML cross-platform delivery strategy. A standard, fully documented file format is also important to permit data exchange between applications running on the same operating system. For instance, the 3D file format supported by QuickDraw 3D provides a means of sharing 3D data between applications and across platforms. QuickTime and QuickTime VR also support a publicly documented file format.
- The technology is *scalable*. The quality of the user's experience when interacting with multimedia content should depend primarily on the capabilities of the user's hardware, not on the limitations of the multimedia content or playback software. All the main QTML technologies can take full advantage of the available memory or other hardware on the user's computer. For example, QuickDraw 3D automatically uses any available supported 3D acceleration card to speed rendering and other operations. Similarly, a QuickTime VR panorama can be viewed at differing resolutions, depending on the amount of available RAM. QuickTime can take advantage of multiple processors on the MacOS to speed its calculations. This scalability is another facet of the “author once, deliver many” philosophy: the same data file can provide vastly different experiences, depending on the available hardware.

In short, the QuickTime Media Layer provides a platform-independent standard for the creation, distribution, and playback of digital media, including video, sound, rendered objects, immersive panoramas, and manipulable objects.

## THE STARS

Now let's take a brief look at the three principal parts of the QuickTime Media Layer: QuickTime, QuickDraw 3D, and QuickTime VR.

### QuickTime

QuickTime is the core of the QTML. It provides a cross-platform multimedia architecture that allows integration of a wide variety of media data types, including graphics, sound, video, text, music, 3D objects, and sprites — with the ability to synchronize all these media types to a common time base. In a word (or two), QuickTime manages time-based data. A collection of time-based data is called a *movie*. QuickTime provides tools to display movies and to let the user interact with movies in appropriate ways (starting, stopping, pausing, and so forth). It also provides the capability to interact with movie data in other ways as well (compressing, expanding, cutting, pasting, copying, and so forth).

In the years since it was introduced, QuickTime has gradually added support for a number of media data types. For instance, QuickTime version 2.0 added support for the QuickTime Music Architecture (more on that later) and version 2.5 added support for MPEG-encoded video. Current versions of QuickTime support 3D data and sprite tracks. It's fairly easy to add support for a new data type because QuickTime is built on a component architecture. (A component is a piece of code managed by the Component Manager that provides a defined set of services to one or more clients.) Each QuickTime component provides an interface to a set of features associated with the manipulation of some sort of data (which might or might not be time based).

The latest version of QuickTime for both MacOS and Windows machines is QuickTime 3.0, which provides several important advancements over previous versions, including expanded file format support, a media abstraction layer, and accelerated visual effects:

- QuickTime 3.0 supports playback, editing, and integration of QuickTime data, MPEG files, AVI, OMF, DVCAM, and OpenDML files, thereby providing one of the highest levels of integration with all major video file formats. QuickTime 3.0 also supports a wide variety of sound file formats, including Wave, AIFF, AU, MPEG Layer 2, and MIDI formats.
- The new media abstraction layer provides QuickTime with a means of accessing hardware accelerators or other multimedia enhancements in a way that is transparent to the software using the QuickTime API. This ensures that existing applications will benefit from these enhancements without any changes.
- QuickTime 3.0 includes enhancements to the QuickTime software architecture that standardize the way in which applications work with visual effects and transitions. For example, QuickTime 3.0 includes a large set of built-in software-based effects, such as cross-fades, chroma keying, SMPTE wipes, and color adjustments.



From the programmer's point of view, it's relatively easy to add support for QuickTime movies to an application. With a few lines of code, you can open a movie file and provide a standard user interface for the user to control the movie playback. The movie file itself can contain all the data needed to synchronize the various data types displayed in the movie. QuickTime also provides a large set of functions for creating and editing movie data.

### QuickDraw 3D

QuickDraw 3D is a cross-platform graphics library that you can use to create, configure, and render 3D models. You can also use QuickDraw 3D to manage user interaction with a rendered 3D scene, such as navigating within the scene (that is, changing the camera angles) and selecting objects in the scene. QuickDraw 3D supports a wide range of basic geometric objects and transformations of objects, as well as attributes for those objects. QuickDraw 3D also supplies several lighting models, shaders, and renderers.

The QuickDraw 3D graphics library supports a C-based API. Most of the API provides a standard object-oriented approach to 3D graphics, wherein you create objects that can inherit properties and behaviors from other objects. For applications that require only the display of 3D objects and limited user interaction with those objects, QuickDraw 3D also supplies a high-level API for the 3D Viewer. In a sense, using the 3D Viewer is like using the standard movie controller to display QuickTime movies: it's very easy to provide a standard interface to the underlying data.

Like QuickTime, QuickDraw 3D is extensible, though not in precisely the same manner. QuickDraw 3D does not support a component-based architecture; instead, it allows developers to extend its capabilities by defining custom objects. Moreover, QuickDraw 3D supports a hardware abstraction layer — called the QuickDraw 3D Rendering and Acceleration Virtual Engine (RAVE) — that allows for plug-and-play hardware acceleration.

QuickDraw 3D also defines a platform-independent file format, called the 3D Metafile Format (3DMF), for storing and interchanging 3D data. This format is intended to provide a standard format according to which applications can read and write 3D data (even applications that do not use QuickDraw 3D to render images). QuickDraw 3D supplies functions that you can use to read and write data in 3DMF files.

### QuickTime VR

The new kid on the QTML block is QuickTime VR, an imaging technology that allows users to interactively explore and examine photorealistic, three-dimensional virtual worlds and objects. QuickTime VR is really two separate technologies in one package. One part of QuickTime VR supports panoramic nodes (or "panoramas"), where the viewer can turn around, as if sitting on a rotating stool, to view different parts of the space around him or her. The other part of QuickTime VR supports object nodes (or "objects"), where the viewer can turn an object horizontally and vertically, as if picking it up and examining it. Any number of panoramas and objects can be linked together into a scene. Clicking predefined areas in a particular node ("hot spots") can move the viewer to another node in the scene or initiate other actions.

QuickTime VR is like QuickDraw 3D in that both technologies are geared toward spatial data. Both of them try, in different ways, to make it seem as if you're in a spatial location, populated by 3D objects. QuickDraw 3D is a traditional 3D graphics library, where each and every object in a scene must be described geometrically and rendered in real time as the viewer's location changes in 3D space. QuickTime VR, on the other hand, works with data that is typically captured photographically and hence can provide substantial detail with a very small data size.

QuickTime VR playback has been available on both MacOS and Windows machines for several years. Early this year, Apple introduced a C language API for controlling VR movie playback. See "Programming With QuickTime™ VR" in this issue for a detailed description of that API.

### THE SUPPORTING CAST

The QuickTime Media Layer is associated with several other important technologies. Some of these are really part of the QTML but deserve special mention, and some are not strictly part of the QTML but provide some nifty capabilities when used with it. Some of these QTML-wannabees are not yet available cross platform, however.

#### Sound Manager

The Sound Manager is the part of the QuickTime Media Layer that manages sounds. For instance, when you play a QuickTime movie, it's the Sound Manager that is ultimately responsible for turning the audio data included in the movie into sounds. This process might involve a good bit of work. For example, the audio data might have to be uncompressed; the uncompressed data might then require that its playback rate be changed; the rate-shifted data might then have to have its volume adjusted; finally, the audio data might have to be mixed with other sounds already playing.

The Sound Manager is available on computers running both the MacOS and Windows operating systems. Like QuickTime, its operations are handled by a variety of components, so it is relatively easy to add capabilities (for instance, to handle different compression and expansion algorithms) to the Sound Manager by writing your own custom components.

#### QuickTime Music Architecture

The QuickTime Music Architecture (QTMA) was introduced as part of QuickTime 2.0. It provides an interface to MIDI, a standard music and device-control architecture, but does not require that any actual MIDI devices be attached to the computer. The QTMA can play individual notes and sequences of notes (generated on the fly or prerecorded) on any available MIDI device, or on a software-based MIDI synthesizer if no external devices are available. You can also use the QTMA to read input from external MIDI devices.

One advantage of the QTMA is that the amount of data required to generate a tune is significantly smaller than that amount of data contained in a digitized recording of that tune. If music figures importantly in your multimedia content, you should consider the QTMA as the delivery vehicle.



## SoundSprocket

SoundSprocket is the part of Apple's Game Sprockets that provides 3D filtering for sounds. (See [Vineyard 1997] for more detail about the Apple Game Sprockets package.) You can use SoundSprocket to make a sound appear to emanate from a specific point in space, and you can change the location of the sound dynamically. These capabilities are especially useful for the QTML components that support a spatial medium, namely QuickDraw 3D and QuickTime VR. For instance, you can assign specific sounds to locations in a panorama; as the user pans or tilts to change the view angle, the location of the sounds appears to change as well.

SoundSprocket is not officially part of the QTML, and currently it's available only on PowerPC-based MacOS computers. Happily, however, it's fairly easy to duplicate some of the SoundSprocket functionality using QuickTime or the Sound Manager. You can simulate 3D filtering of sounds for QuickTime movies by adjusting the balance and volume of a movie's sound track as the user changes spatial positions or orientations in a 3D scene or a VR panorama. Also, you can simulate 3D filtering of sounds played using the Sound Manager by issuing the volumeCmd sound command, which controls the volume and balance of the left and right speakers independently.

## PlainTalk

Apple provides two speech technologies bundled together under the general name PlainTalk: speech synthesis and speech recognition. Speech synthesis is the process of converting written tokens (text) to spoken tokens (speech). This can be useful to provide a narration of a walk-through or to vocalize a QuickTime text track. Speech recognition is process of converting spoken words into recognized utterances. This is useful to give the user another input method. For instance, instead of having the user pan and tilt in a VR panorama using the mouse or keyboard, you can support speech commands to achieve the same effect. (See [Pallakoff and Reeves 1996] and [Monroe 1996] for several good articles describing Apple's speech recognition capabilities.)

Once again, these technologies are not officially part of the QTML, but they can dramatically enhance the user experience when combined with the multimedia technologies provided by the QTML.

## QuickTime Conferencing

QuickTime Conferencing (QTC) is a set of software components that support sharing time-based media across local- and wide-area networks. In other words, QTC provides real-time multimedia communications. You could use QTC, for example, to support videoconferencing or a "virtual whiteboard." QTC provides a number of components for managing the network interface and other operations, and also uses standard QuickTime components when possible.

## INTEGRATION

The real fun with the QuickTime Media Layer is making it all work together. Each component of the QTML has a programming interface, so you can combine components simply by using the APIs together. One of my favorite early examples of QTML

integration is Robert Dierkes' TextureEyes application, which can texture map a QuickTime movie (or even a live video feed!) onto a QuickDraw 3D object. You could use the same technique to play a movie on a rendered TV screen in a 3D scene. And then you could take that rendered object, with its QuickTime movie texture, and embed it in a QuickTime VR panorama. And then, with a few simple Movie Toolbox functions, you could adjust the sound balance and make the movie's sound track get louder or quieter as the user pans toward or away from the TV set.

Similarly, it's very easy to integrate SoundSprocket and QuickDraw 3D to attach sounds to specific locations in a rendered 3D scene. (Indeed, the SoundSprocket API uses many QuickDraw 3D data structures to describe the location and orientation of the virtual listener and the sound sources.) It's also reasonably easy to attach sounds to specific locations in a QuickTime VR panorama and to link the orientation of the SoundSprocket listener to the current orientation of the viewer in the panorama.

## CONCLUSIONS

The QuickTime Media Layer is a set of cross-platform Apple technologies that support the authoring, delivery, and playback of multimedia content. Used together, these technologies provide a means to integrate spatial and temporal data into a rich, unified, interactive user experience. The data underlying this experience can be generated dynamically, or read from files stored locally, on CD-ROM, or on a remote server accessed across a network.


It's important to keep in mind that the QTML is a unifying *strategy*, not a finished product. At the current time, true media integration must be done at the API level or using an authoring environment that supports the various media types. Currently, there is no single, unified file format for the many data types supported by the stars and supporting cast members of the QTML. Part of what this means is that there is no easy way (again, short of programming or using an authoring environment) to animate 3D objects in a QuickTime VR scene or to attach sounds to locations in a 3D scene.

Nonetheless, it's only a matter of time before software developers begin to use the existing APIs to create authoring and playback tools to provide a more seamless integration among all members of the QTML. At that point, the QTML will move outside the ranks of programmers and become the unified media authoring and playback layer for the rest of us.

## BIBLIOGRAPHY AND REFERENCES

Monroe, Tim. "Adding Speech Recognition to an Application Framework". *develop, The Apple Technical Journal*, issue 27 (September 1996), pp. 22-33. Apple Computer's Developer Press.

Pallakoff, Matt, and Arlo Reeves. "The Speech Recognition Manager Revealed". *develop, The Apple Technical Journal*, issue 27 (September 1996), pp. 6-21. Apple Computer's Developer Press.

Vineyard, Jeremy. "Sprockets are Forever". *MacTech Magazine*, 13:2 (February 1997), pp. 12-15. 



by Michael Rutman, independent consultant

# Rhapsody FAQ's

## *You've heard the rumors, but how much is true?*

### **BUT A FRIEND TOLD ME...**

With today's ever changing technology, finding out information about future technologies can be challenging, especially when it is difficult to get hands-on experience with that technology. Rhapsody is no exception. Even though Rhapsody is based on OPENSTEP, which has been out for almost a decade, most developers have not had a chance to play with it themselves.

Most developers rely on word of mouth and trade magazines for obtaining their information. While trade magazines are great for getting facts, word of mouth is what most people rely on. Unfortunately, word of mouth isn't always as reliable as one would hope.

After seeing these technologies in action at WWDC and talking with Apple engineers and Evangelists, many of the rumors are clearly wrong. In this article, I hope to cover popular rumors and debunk them one by one.

### **Display Postscript is slow**

OPENSTEP uses the Display Postscript model for all drawing. This allows true WYSIWIG and easy printing code. OK, sounds good, but aren't there downsides? Isn't my Postscript LaserWriter very slow?

Yes, Postscript used to be very slow, but was so powerful it was worth the wait. Waiting for a high quality printout is not as painful as waiting for the screen to update. When NeXT decided to use PostScript for their screen display, Adobe and NeXT realized they would need to improve PostScript's speed.

The optimizations they implemented included using integer font metrics for screen fonts, lazy depth promotions, compression of the backing store of a window, better algorithms for color conversion and dithering, file mapping, and many more.

So, did this work? In a nutshell, yes, it worked very well. As Apple is proud of saying, Display Postscript is 266 times as fast as the LaserWriter NT. So, if you believe Apple statistics, and if your LaserWriter NT can do 16 pages a minute, then you should be able to do 71 frames a second, which is as fast as the monitor can display anyway. It is nice what you can do with statistics, but how does it feel?

To prove that Display PostScript is quite nice, Apple put up two windows running different QuickTime movies with a lot of motion. During their live demo, they dragged the windows around the screen overlapping the two windows; the audience saw perfect clipping, no stutter, no delays. The QuickTime movies continued playing as they were dragged. Most Macintosh users are used to dragging an outline of a window instead of a window. MacOS drags outlines for speed. Under Rhapsody, Display Postscript is fast enough to continue displaying QuickTime movies while being dragged.

Seeing is believing, and I saw Display Postscript displaying a lot of nice graphics very quickly.

### **I need to know postscript to program under Rhapsody**

Rhapsody draws in PostScript, and knowing PostScript would be useful. However, there is a difference between it being a good idea to know PostScript and having to know PostScript.

The vast majority of PostScript is hidden from the developer. OPENSTEP's framework, called the AppKit, handles all the setup

**Michael Rutman** is a software developer with experience developing for several platforms, including Macintosh, NeXTSTEP, Newton, Pilot, and Windows NT. While working at Software Ventures, he lead the development of Snatcher and MicroPhone Pro for NeXTSTEP. He also worked on the MicroPhone Pro for Macintosh product line. He now works as an independent consultant on a variety of projects including encryption, compilers, web based add-rotation software, and ship stevedoring. To contact Michael Rutman send mail to [moose@manicmoose.com](mailto:moose@manicmoose.com).



PostScript needs for drawing or printing views. A developer that wants to modify the default behavior would have to know PostScript, but the vast majority of developers will never need to do this.

The common PostScript commands — the ones for drawing lines, circles, rects, setting colors, and so on — are wrapped by C functions. Drawing a rectangle is done with `NXFillRect( const NXRect *)`. A developer need only pass a rectangle to this routine and the rectangle is drawn. Even though the low level drawing is done in PostScript, the developer need only know that there is a routine that draws a rectangle. For finer control, `PSsetColor(NXColor *)` sets the drawing color, `PSmoveto`, `PSlineto`, and `PSStroke` can do individual lines.

Almost all entities displayed, such as boxes, buttons, popups, and windows are objects that hide all of the PostScript needed for drawing. As long as standard controls are used, no PostScript need be created. Of course, if a complex PostScript entity has to be generated, the code can be written into a PSwrap and downloaded to the PostScript server. However, to do this one would have to learn PostScript. My experience is that pswraps are useful, but rarely necessary.

### **My mom isn't going to want to learn UNIX**

Rhapsody is based on UNIX, but what does that mean? Many people believe that UNIX is a command line interface into a file system where you have to type `grep` a lot. Nobody wants to dump this on an unsuspecting user, and NeXT has worked on hiding UNIX for years. Now that NeXT and Apple are combined, there is a strong push to hide UNIX even farther.

How to hide UNIX has always been tricky, and early versions of NEXTSTEP didn't do a very good job of it. Because UNIX kept peeking up its ugly head in the early days, there is a lot of word of mouth bashing of Rhapsody, and that is unfortunate. Over the last 8 years, NeXT has done a good job of putting the UNIX layer well under a pleasant user experience. Everything your mom can do on the Mac can be done under Rhapsody in the same way.

OPENSTEP, however, is for both your mom and you. For power users, OPENSTEP still has a terminal window that accesses the UNIX layer directly, and those developers who want to can type `grep` to their hearts content. If they don't want UNIX, there are other ways to do what they want to accomplish. The UNIX command line is there only for the people that want it. It is important to remember the difference between having a tool available to you, versus having to use a particular tool to get your job done. Rhapsody is offering a choice.

### **Game developers won't be able to make their software work**

Game developers program to the hardware. The life of a game is measured in months, not years, so working with the next system upgrade is a low priority, but an extra frame a second can make or break a game. To get that extra frame a second, game developers need access to the hardware, and the Rhapsody framework hides the hardware.

The Rhapsody engineers understand this problem, and they want to play games as much as the rest of us. They have placed hooks in at the lowest levels called interceptors.

Interceptors allow game developers to write code that accesses the monitors at the lowest levels, completely bypassing anything that might slow down a game. Game developers that need to take over an entire monitor and blit bits around can grab the monitor through an interceptor.

### **It won't look like the Macintosh I've come to love**

This is the opinion that is mostly contested. Rhapsody is going to look like MacOS 8, not Mac 7.5. How different are the two is still up for debate. MacOS 8 has a more "modern" look and feel, but it also is still in development as I write this article. Some have argued that the "modern" look and feel is unpleasant, but it is still being cleaned up, so it is hard to judge.

On the other hand, the Macintosh look and feel has been evolving since 1984, and current Macintoshes do not look much like they did 13 years ago. The argument that the latest system doesn't look like the Macintosh I've come to love could have been made years ago, and we would not have a lot of the little things that make the Macintosh fun. Color is a good example. When Apple added color to the Macintosh, it changed the look and feel. Most people now agree that color was a good change.

So, it won't look like the Macintosh you bought 5 years ago, then again, neither will any other Macintosh you buy next year, whether it is running Rhapsody or not.

### **It's not a Macintosh, no matter how much it looks like a Mac**

This is an interesting argument, and it goes along the lines of: The Macintosh is what I have on my desk, and what I've learned to program. If you go to UNIX, and change the Toolbox, then it isn't a Macintosh. Since it won't be a Macintosh, it won't work like a Macintosh. Since the Macintosh is easy to use, and this won't be a Macintosh, it will be hard to use. Believe it or not, I've heard that argument from people that should have known better. Despite all the flaws in the argument, it's an emotional fight, and those are always tricky.

This is my feeling about whether Rhapsody will be a Macintosh. If it looks like a duck, walks like a duck and quacks like a duck, then it's a duck. In the same vein, if it looks like a Macintosh, runs my Macintosh software, and follows all the Macintosh guidelines, then it's a Macintosh.

Now, I'll be the first to admit, change is bad and should not happen, but in the computer industry, change is inevitable. The Macintosh look and feel is evolving, and those changes are going to effect us. Most of the changes to the look and feel are actually going to appear in MacOS 8, (code named Tempo) not Rhapsody.

### **Programming Rhapsody is so easy, we won't need programmers anymore**

In a way, there is a grain of truth to not needing programmers. Traditionally, creating a user interface required code changes. Each item in each window needed hard coded locations. Changing a button title or position could require a code change and a recompile. Some dialogs could be edited with ResEdit, but ResEdit is hardly a tool for novices.



Under Rhapsody, interfaces are completely separated from the code. Editing User Interfaces is done through a program called Interface Builder. Interface Builder has a lock mode where non-programmers can safely make changes to a UI without breaking links. On many of my OPENSTEP applications, non-programmers have done the final cleaning up of my user interface files.

Separating the UI from the coding usually cuts the need for a programmer in half. This is not the same as not needing a programmer, because someone has to make the program useful. The graphic artist can then make the program usable without taking up the programmer's time. The reality is that making a software package useful as opposed to pretty, still needs a programmer.

### **None of my Macintosh programs are going to work under Rhapsody**

Again, there is a grain of truth here, but not much more than a grain. Most code written to the Toolbox must be rewritten to take advantage of the new Rhapsody world. Even though programming for Rhapsody is easier, getting developers to rewrite hundreds of thousand of lines of code is not likely.

Transitions to a new architecture can be painful for users. Fortunately, Apple went through this when they moved from 68K machines to PowerPC. Even though few existing applications "worked" in the native PPC, the 68K emulator allowed almost all existing software to run on the PPC, even though it wasn't native. This allowed some people to upgrade, running their software without abandoning all their existing software. Today, some applications being shipped work only on PPC.

In a very similar way, Apple has provided a mode under Rhapsody that will allow users to run old applications. Currently, this is called the Blue Box, but over the next 18 months Apple marketing may decide on a more user friendly name.

The Blue Box has two modes: one puts Blue Box in a Rhapsody window, the other lets the Blue Box take over the entire display space. In the latter, the machine looks and acts like a traditional Macintosh, while Rhapsody applications continue to run underneath the UI. The user experience is almost the same as a machine running MacOS 8, but having Rhapsody underneath adds stability for Rhapsody applications.

When Blue Box is run in a Rhapsody window, all of the Blue Box applications will put their windows inside the Blue Box window. While switching between window and full screen mode is easy, the overall experience is ugly. At least it allows users to run their one or two Blue Box apps while living in a Rhapsody world.

### **Rhapsody does not support Undo**

Undo is very difficult to implement, and many applications spend more time supporting undo than any other feature. NeXT got a bad rap because none of their existing applications support undo, and many early developers did not support undo.

One of the main reasons for the lack of undo was, ironically enough, OPENSTEP's ease of development. Using OPENSTEP, developers were able to get a large number of new features

added to an application very quickly. Unfortunately, undo is still difficult. This leads to a developer not wanting to take the extra time to add undo for an action where undo can take longer to implement than the feature the developer is adding.

However, Apple recognizes that undo is very important to Macintosh users. Technology for adding undo has advanced, and adding undo is easier than it was years ago. Today, the undo architecture of Rhapsody allows a quick and easy multiple-level undo, once the developer has written undo actions for everything the user can do. Writing the individual undo actions are going to be as hard as they've always been, but that is true regardless of the platform.

### **I'll need to learn a new language**

OPENSTEP has used Objective-C for many years, and the framework is designed for access from Objective-C. Many years ago, NeXT realized that many programmers needed to use C++, and OPENSTEP had to support C++. Unfortunately, the C++ object model and the Objective-C object model do not mesh very well, and C++ has always been a second class citizen.

The reason C++ had to take a back seat is because Objective-C allows run-time binding, and without it, C++ is just not powerful enough to fully access the AppKit without using Objective-C. NeXT provided a way of using both Objective-C and C++ in the same routines. This allows a developer to do his back end in C++ and his UI in Objective-C. Overall, developers were happy, but many complained about needing to use two different languages.

Then along came Java. Java is like C++ in syntax, but has the run-time binding that Objective-C needs. Apple found that Java's object model is very close to the Objective-C object model, and set up the Java Virtual machine to allow code written in Java to access the Objective-C framework directly. Even though the developer coding in Java is calling Objective-C objects, everything is transparent. For the Java developer, programming the AppKit is done entirely in Java. At least that's the theory, I haven't had a chance to try it myself.

Once the decision was made to allow the Java developer to access the AppKit directly, the next logical step was to make Objective-C more like Java. Having developed in Objective-C for many years, I, personally, do not see why developers would want a Java syntax, but most developers I've talked to have made it clear that they would rather use the Java syntax than the Objective-C syntax. Both are supported, so old time developers can continue to use the old syntax, and new developers only have to know the syntax of Java.

Now, the new Objective-C syntax may have a Java-like syntax, but it isn't Java. It's still Objective-C, which means you still have all the power of C. One advantage is that developers can jump between C++, Java and Objective-C as the need arrives. Taking a C++ library, hooking it up to some Java networking code, and throwing on an Objective-C interface is easy, and each language can directly access any of the others. With the new syntax for Objective-C, everything will have roughly the same look.



### **I'll need all new drivers for my hardware**

Yes and no. One of the goals of Rhapsody is making the drivers more standard. SCSI drivers, for example, have always been a problem on the Macintosh. Personally, I have 3 different drivers for removable media, and none is compatible with any of the others. Under Rhapsody, I don't need special drivers for most SCSI hardware. It would be nice if all drivers worked this way, unfortunately, not all drivers will work and those drivers must be updated.

Because Apple recognizes that their users will need to continue using their existing devices, Rhapsody has a dual boot mechanism. The Macintosh can be booted into Rhapsody and run the Blue Box, but if a device is only going to work under traditional Macintosh, then the machine can be rebooted to run the traditional Macintosh OS. In this mode, Rhapsody won't be running at all — all old drivers should then work.

Fortunately, the new driver kit makes writing drivers easier. Apple knows that Rhapsody will only succeed if users can use their devices, and Apple is working with developers to make sure they are using the new driver kit. Developers should be happy with the new driver kit as it will cut development time dramatically.

### **Apple's abandoning all these cool technologies**

First, Apple has not abandoned the technologies we have come to rely on. They have put some of their technologies in maintenance mode, which means there aren't going to be any wizzy new features added to it. Some bug fixes, and system upgrades will still happen.

One example of an "abandoned" technology is OpenDoc. The rumors are that nobody is working on OpenDoc, and all developers that committed to it are now out of business. The reality is that OpenDoc has been ported to MacOS 8, and is already running on the Blue Box under Rhapsody. All existing OpenDoc parts will run under Rhapsody in the Blue Box.

Furthermore, even though Apple has no announced plans to port OpenDoc to be a native Rhapsody application, Apple is not stupid. If a market appears for OpenDoc parts, Apple still has all the source code, and they still have engineers that know the internals of OpenDoc. If Apple finds out that another technology they created is taking off and becoming an industry standard, I would expect Apple to jump right back in. Developers that have committed to OpenDoc may not have the huge market they were hoping for, but if they can create the market, Apple will be right back on board. Now, that's just my opinion, and it is unlikely that Apple will comment one way or another, but I think it's a safe bet that Apple will do what's in their best interest.

Justification of this opinion, however, can be seen in Game Sprockets. Less than 2 months ago, it was announced that Game Sprockets would be in maintenance mode. By WWDC, Apple was shown that Game Sprockets is an emerging standard, will help sell machines, and is just a generally good idea to have. Game Sprockets are back in development, and are being ported to Rhapsody.

### **Whatever Apple announces at WWDC is dead in a year**

In the past, Apple has shown developers new technologies as early as possible, and the developers helped shape each technology. By the time the technologies were ready for users, the technologies often had very little in common with what Apple first presented. Occasionally, once the technology had been hashed out with developers, Apple realized that it may be cool, but nobody in the market cares. They would then cancel the technology, and the developers that wanted to use the technology were, justifiably, upset.

This year, there is a difference. Apple is not showing us technology in its infancy, but technology that has been ported to several platforms and used for many years. The main thrust of WWDC was Rhapsody and how all the other parts of Apple will fit into Rhapsody. Some of the technologies Apple showed at WWDC are in their infancy, and I would expect changes in those technologies, and I would not be surprised if a few of the things I heard never saw the light of day. Rhapsody, fortunately, is unlikely to be canceled.

### **Macintosh Developers have abandoned the Mac and are all developing Windows apps**

Every year I hear this, and every year I look around and see thousands of Macintosh developers. True, there are many developers that have moved to Windows, and a lot that have moved to Java, but there are also many new Macintosh programmers. WWDC had almost 2000 developers, and that's almost as many as they had last year. Of course, last year, there were additional developers that only attended for one or two days, so it's harder to compare.

Companies, however, have had to make a business decision. Many companies last year announced a wait and see attitude on the Macintosh. Some companies even downsized their Macintosh departments. The good news is that other companies jumped into the market to fill the void created by those companies jumping ship.

Interestingly enough, some of the companies that downsized their Macintosh departments have been struggling to rehire Macintosh programmers, usually at a higher rate than they were paying. It turns out that 27 million Macintosh users world-wide have a large influence on purchasing software. Companies that thought the Mac market was dead have had to get back into the market quickly to meet the demand that Macintosh users are creating.

### **Apple has lost market share**

Yes, Apple has lost market share, but the MacOS has gained market share in the last year. The clone makers have taken over a major chunk of the Macintosh market. Some have worried that the clone makers are cannibalizing Apple's share, and this has happened to some degree, but they have also helped Apple expand the market. In reality, Apple has seen the market share of the MacOS grow by 11%.



## Apple engineers are all depressed and looking to quit

When Apple laid off over 4000 employees and contractors, the mood at Apple was less than happy. However, like all good programmers, once they had an objective, their focus and happiness returned. Just about every Apple engineer is excited about their new system, and all of them are happy to spread the word. They are cranking out Rhapsody in record time, and at WWDC, they raced to get their latest creations into the show. Every engineer that talked was excited about both Rhapsody as a whole, and their individual efforts.

## Apple always talks Pie in the Sky and promises a lot


Yes, they do. This time, for a change, from the CEO on down, the goal was to under promise and over deliver. On this, they did a great job. Several times, a developer would recommend an improvement, and almost every time the answer was, "That's great, we want to do it, but right now, this is what we have to deliver. Maybe next year." After having gotten used to a company that has, more often than not, bitten off more than it can chew, it was wonderful seeing a team that could stay focused on what they are working on and leave the dream solution for later day.

This is not to say that Apple did not recognize developer concerns, quite the opposite. When a developer brought up a point, the Apple engineers considered it, and where they could, they talked about how a developer would deal with that problem. In some cases, Apple agreed to make changes, in others, Apple explained why they made the decisions they did. They have a lot of bright people working on these problems, and they have a lot of open minds.

The most refreshing part, by far, was the Apple engineers admitting that they can deliver a very good system without it being the end-all system. They admit that there will be problems to be worked out, but that won't stop them from delivering a good system in a reasonable time frame. The best part, most developers understood and respected Apple engineers when they said "Not this year."

## HOW DO I FEEL ABOUT THINGS?

I'll admit that I was pretty excited about Apple merging with NeXT before WWDC, but what I got was better than I could have ever hoped. I had been afraid that Apple engineers were dejected, NeXT engineers would gloat, and nobody would know what they were going to deliver. Instead, I saw Apple and NeXT engineers working together, everyone excited, and some pretty cool stuff being developed.

If you weren't there, you wouldn't have seen the glow in the engineer's eyes when they had the MacOS (Blue Box) running on Rhapsody. You wouldn't have known that they feel they are creating something special. Most of all, you wouldn't have seen a management team listening to the developers, and taking notes to suggestions. 


## Viewpoint, continued from page 4

much of the user interface code is already written, we developers are free to ship sooner or spend time adding more features.

## Still Some Holes in Rhapsody

Rhapsody is not all peaches and cream. There are some holes; the poor networking support is the biggest. Between Apple's presentations and my conversations in the halls of WWDC, it is clear that Apple still doesn't understand how the Internet is changing; their "strategy" is limited to web content development and deployment. They seem to be ignoring many other areas including communications, collaboration and commerce. For example, Yellow Box does not provide any foundation classes for using common Internet protocols such as HTTP or FTP. It doesn't even provide a foundation class for basic IP networking. This means that every network-savvy Yellow Box application must use platform-specific network code. A Rhapsody application will use Rhapsody's version of BSD sockets, a MacOS application will use OpenTransport streams and a Wintel application will use WinSock. Given that networking is the fastest growing segment of the software industry, it seems absurd that Apple does not provide a network abstraction layer in Yellow Box as they have for the user interface.

This networking issue also raises the question of why Apple is using sockets instead of streams. Everyone I spoke with has made a strong argument for streams being a more flexible and generally superior architecture. When Apple asked us all to move our network code to OpenTransport (streams), they said it was a technology for the future. Streams provides standard APIs at all levels, allowing developers to write code to adjust the contents of the network at the level that makes sense for their software. Some developers argued that Apple should develop a sockets interface to make porting code to other platforms easier. Apple answered that sockets could easily be implemented on top of streams, and some third parties offered libraries doing exactly that. It is very hard to go the other way because sockets implementations really define only the highest level API. The underpinnings are specific to each platform.

Not only is it odd that Apple seems to have ignored their original reasons for moving all developers to a streams implementation, but Apple already has most of the streams code for Rhapsody, including complete network stacks for AppleTalk, IP, IPX, and TokenTalk. Instead, they are porting the NeXTSTEP sockets code and adding the AIX AppleTalk sockets code; then they have to find a way to make Blue Box's OpenTransport (streams) work on top of this. Many WWDC attendees asked Apple how they intended to solve specific, difficult problems with this scenario. Apple's most common answer was "Hm, we hadn't thought of that. That's a hard problem, but we're sure we can solve it when we get to it." To Apple's credit, they repeatedly took some harsh criticism from attendees, and they appeared to really listen. Hopefully they got the message that they should reevaluate the technical merits of choosing sockets over streams so we can have a networking architecture that can grow with the rest of the platform. 



by Bob Boonstra, Westford, MA



## DISAMBIGUATOR

The Challenge this month is to write a string completion routine loosely patterned after the keyword lookup facility in the QuickView utility. QuickView will suggest a completion of the keyword as you begin to type it, and update that suggested completion as you continue to type. In the Toolbox Assistant, for example, if you are looking for documentation on `InitGraf` and type "i", the suggested completion is "iconIDToRgn". As you continue by typing "n", the suggestion becomes "index2Color". Adding "i" yields "initAllPacks"; adding "t" leaves the suggestion intact; adding "g" changes it to "initGDevice". Finally, typing "r" gives the desired "initgraf".

For our disambiguator, you will be given an unsorted list of words and an opportunity to preprocess them. Then you will be given a string to match and asked to return a list of words matching `findString`. To make the problem more interesting, the match string can contain wild card characters, as described below.

The prototype for the code you should write is:

```
typedef unsigned long ulong;

void InitDisambiguator(
    const char *const wordList[], /* words to match against */
    ulong numWords,              /* number of words in wordList */
    void *privStorage,           /* private storage preinitialized to zero */
    ulong storageSize            /* number of bytes of privStorage */
);

ulong /*numMatch*/ Disambiguator(
    const char *const wordList[], /* words to match against */
    ulong numWords,              /* number of words in wordList */
    void *privStorage,           /* private storage */
    ulong storageSize,           /* number of bytes of privStorage */
    char *findString,            /* string to match, includes wild cards */
    char *matchList[]            /* return matched words here */
);
```

Your `InitDisambiguator` routine will be called with an unsorted list `wordList` of `numWords` null-terminated words to match. The `wordList` words will include alphanumeric characters, spaces, and underscores. You will also be provided with a pointer `privStorage` to `storageSize` bytes of preallocated memory initialized to zero. The amount of storage provided will be at least 20 bytes for each word in `wordList`, plus one byte for each character in the `wordList` (including the null byte, and rounded up to a multiple of 4). In other words, `storageSize` will be no smaller than `minStorage`, calculated as:

```
for (minStorage=0,i=0; i<numWords; i++)
    minStorage += 20 + 4*(1+strlen(wordList[i])/4);
```

`InitDisambiguator` is not allowed to modify the `wordList`, but you may store a sorted version of `wordList`, or pointers to the words in sorted order, in `privStorage`. The first four parameters provided to `Disambiguator` will be identical as those provided to `InitDisambiguator`. In addition, you will be provided with the null-terminated `findString` and a preallocated array `matchList` with `numWords` entries where you are to store pointers to the words that match `findString`. Your string matches should be case insensitive (i.e., "initgr" matches "InitGraf". The `matchList` should be returned with the strings ordered in case-insensitive ASCII order (i.e., space < [0..9] < [A-Za-z] < underscore).

The `findString` may also contain zero or more of the wildcard characters '?', '\*', and '+'. The wildcard '?' matches any single character, '\*' matches zero or more characters, and '+' matches one or more characters. So, for example, "graf" matches any string ending in the (case-insensitive) string "graf", while "+IInd+" matches any string containing "IInd" between the first and last characters of a word.

## THE RULES

Here's how it works: each month we present a new programming challenge. First, write some code that solves the challenge. Second, optimize your code (a lot). Then, submit your solution to MacTech Magazine. We choose a winner based on code correctness, speed, size, and elegance (in that order of importance) as well as the submission date. In the event of multiple equally desirable solutions, we'll choose one winner (with honorable mention, but no prize, given to the runner up). The prize for each month's best solution is a \$100 credit for Developer Depot™.

Unless stated otherwise in the problem statement, the following rules apply: All solutions must be in ANSI compatible C or C++, or in Pascal. We disqualify entries with any assembly in them (except for challenges specifically stating otherwise.) You may call any Macintosh Toolbox routine (e.g., it doesn't matter if you use `NewPtr` instead of `malloc`). We compile all entries into native PowerPC code with compiler options set to enable all available speed optimizations. The development environment to be used for selecting the winner will be stated in the problem. **Limit your code to 60 characters per line** or compress and binhex the

solution; this helps with e-mail gateways and page layout.

We publish the solution and winners for each month's Programmer's Challenge three months later. All submissions must be received by the 1st day of the month printed on the front cover of this issue.

You can get a head start on the Challenge by reading the Programmer's Challenge mailing list. It will be posted to the list on or before the 12th of the preceding month. To join, send an email to [listserv@listmail.xplain.com](mailto:listserv@listmail.xplain.com) with the subject "subscribe challenge-A".

Mark solutions "Attn: Programmer's Challenge Solution" and send it by e-mail to one of the Programmer's Challenge addresses in the "How to Communicate With Us" section on page 2 of this issue. Include the solution, all related files, and your contact info.

MacTech Magazine reserves the right to publish any solution entered in the Programmer's Challenge. Authors grant MacTech Magazine the exclusive right to publish entries without limitation upon submission of each entry. Authors retain copyrights for the code.



For each call to InitDisambiguator, your Disambiguator routine will be called an average of 100 to 1000 times. The winner will be the solution that finds the correct matchList in the minimum amount of time, including the time taken by the initialization routine.

This will be a native PowerPC Challenge, using the latest CodeWarrior environment. Solutions may be coded in C, C++, or Pascal. The problem is based on a suggestion by Charles Kefauver, who pointed me to an April, 1995, *AppleDirections* article discussing the user interface for a disambiguator. Charles wins 2 Challenge points for his suggestion.

### THREE MONTHS AGO WINNER

Congratulations to **ACC Murphy** (Perth, Australia), for submitting the faster (and smaller) of the two entries I received for the Projection Challenge. This problem required contestants to calculate the image of a set of input polygons, including the shadows cast by one polygon on another, given an observation viewpoint and an illumination point.

Both of the submitted solutions used a ray-tracing technique. The winning solution calculates, for each point on the projection plane, the nearest polygon to the viewpoint among those intersecting the ray from the plane to the viewpoint. It then does another ray-trace to determine if there are any other polygons between the illumination point and the projected polygon, identifying the point as being in shadow if an intervening polygon is found.

I ran three test cases, moving the polygons 10 times for a given viewpoint in each case, using a GWorld bounds rectangle slightly smaller than my 1024x768 monitor. As you can see from the execution times, considerable refinement would be needed before this code could be used for animation.

A good discussion of the projection and hidden surface removal algorithms applicable to this problem can be found in the *Black Art of Macintosh Game Programming*, by Kevin Tieskoetter. In addition to discussing the z-buffer ray-tracing algorithm, it describes another technique for hidden surface removal called the Painter's algorithm. This approach breaks the polygons to be displayed into pieces such that each piece is entirely in front of or entirely behind any other piece, as seen from the viewpoint. The polygons can then be sorted and displayed without looking at each pixel in the image. For our application, two polygon decompositions would be required, one for the image, and one for the shadows.

The table below lists, for each entry, the execution time for each case and the code size. The number in parentheses after the entrant's name is the total number of Challenge points earned in all Challenges to date prior to this one.

Name	Case 1 Time	Case 2 Time	Case 3 Time	Total Time (secs)	Code Size
A.C.C. Murphy (10)	29.02	23.64	81.61	134.27	4196
Ernst Munter (232)	20.87	58.11	89.76	168.74	7192

## Bar Code Fonts & Readers

Create bar codes in any Windows or Macintosh program quickly & easily with Rivers Edge bar code fonts.

### Bar Code Fonts

TrueType & Postscript Fonts  
Easy to Use  
30 Day, Money-Back Guarantee  
**\$149 ea**

Choose from:  
**Code 39, Code 93, Code 128,**  
**Int. 2 of 5, Codabar, MSI-Plessey,**  
**PostNet, UPC/EAN/ISBN**

### Bar Code Readers

Attaches to Keyboard  
Plug & Play for Windows/Mac  
Easy to Use

**\$229** - Wand Reader-Basic  
**\$295** - CCD Reader  
**\$495** - Laser Reader



Call for **FREE** Information

**512.219.7768**

fax: 512.219.7769 email: sales@riversedge.com

**www.riversedge.com**

### TOP 20 CONTESTANTS

Here are the Top Contestants for the Programmer's Challenge. The numbers below include points awarded over the 24 most recent contests, including points earned by this month's entrants.

Rank	Name	Points	Rank	Name	Points
1.	Munter, Ernst	194	11.	Beith, Gary	24
2.	Gregg, Xan	114	12.	Cutts, Kevin	21
3.	Cooper, Greg	54	13.	Nicolle, Ludovic	21
4.	Larsson, Gustav	47	14.	Picao, Miguel Cruz	21
5.	Lengyel, Eric	40	15.	Brown, Jorg	20
6.	Boring, Randy	37	16.	Gundrum, Eric	20
7.	Mallett, Jeff	37	17.	Higgins, Charles	20
8.	Lewis, Peter	32	18.	Kasparian, Raffi	20
9.	Murphy, ACC	30	19.	Slezak, Ken	20
10.	Antoniewicz, Andy	24	20.	Studer, Thomas	20



There are three ways to earn points: (1) scoring in the top 5 of any Challenge, (2) being the first person to find a bug in a published winning solution or, (3) being the first person to suggest a Challenge that I use. The points you can win are:

1st place .....20 points	5th place .....2 points
2nd place .....10 points	finding bug .....2 points
3rd place .....7 points	suggesting Challenge...2 points
4th place.....4 points	

Here is A.C.C. Murphy's winning solution:

## Challenge.p

A.C.C. Murphy

```
unit Challenge;
```

```
(*
```

### Assumptions:

Storage space must be big enough for 13 floats per polygon  
All points must be significantly smaller in magnitude than BIG\_FLOAT = 1000000.0  
Polygons are translucent (their colour based upon lighting is independent of the side of the polygon that is lit)  
50% attenuation of colour is used  
50% attenuation of black is black

### Method:

InitProjection is not used

First we precalculate a small bounding sphere for the polygon points.  
Next we get the information about the GWorld to allow direct pixel access.  
Then for each point on the GWorld, we trace the ray from the point to the eye, intersecting it with each polygon and finding the one closest to the eye (furthest forward, since the eye is in front of all polygons). That determines the colour. We then trace the ray from that intersection point to the light source to determine whether the point is in shadow, and if so we halve the intensity. We set the colour of the pixel and move on.

### Optimizations:

Direct pixel access to the GWorld (known to be 32 bit)  
Bounding sphere used to optimize the ray/polygon intersection test.  
Time is approximately 2 microseconds per pixel per polygon on an 8500.

```
*)
```

```
interface
```

```
uses
```

```
Types, Quickdraw, QDOffscreen;
```

```
const
```

```
kMAXPOINTS = 10;
```

```
const
```

```
BIG_FLOAT = 1000000.0;
```

```
type
```

```
float = real;
```

```
type
```

```
My2DPoint = record (* point in z==0 plane*)
```

```
x2D: float; (* x coordinate*)
```

```
y2D: float; (* y coordinate*)
```

```
end;
```

```
My3DPoint = record
```

```
x3D: float; (* x coordinate*)
```

```
y3D: float; (* y coordinate*)
```

```
z3D: float; (* z coordinate*)
```

```
end;
```

```
My3DDirection = record
```

```
thetaX: float; (* angle in radians*)
```

```
thetaY: float; (* angle in radians*)
```

```
thetaZ: float; (* angle in radians*)
end;
MyPlane = record
planeNormal: My3DDirection; (* normal vector to plane*)
planeOrigin: My3DPoint; (* origin of plane in 3D space*)
end;
MyPolygon = record
numPoints: longint; (* number of points in polygon*)
thePoint: array[0..kMAXPOINTS-1] of My2DPoint; (* polygon in z==0 plane*)
polyPlane: MyPlane; (* rotate/translate z==0 plane to this plane*)
polyColor: RGBColor; (* the color to draw this polygon*)
end;
MyPolygonArray = array[0..0] of MyPolygon;
```

```
procedure InitProjection(
```

```
const viewPoint: My3DPoint; (* viewpoint from which to project*)
```

```
const illumPoint: My3DPoint; (* viewpoint from which to draw shadow*)
```

```
storage: univ Ptr; (* auxiliary storage preallocated for your use*)
```

```
storageSize: longint (* number of bytes of storage*)
```

```
);
```

```
procedure CalcProjection(
```

```
offScreen: GWorldPtr; (* GWorld to draw projection *)
```

```
const thePolys: MyPolygonArray; (* polygons to project *)
```

```
numPolys: longint; (* number of polygons to project *)
```

```
const viewPoint: My3DPoint; (* viewpoint from which to project *)
```

```
const illumPoint: My3DPoint; (* illumination point from which to draw shadow *)
```

```
storage: univ Ptr; (* auxiliary storage preallocated for your use*)
```

```
storageSize: longint (* number of bytes of storage*)
```

```
);
```

```
implementation
```

```
type
```

```
Ray3D = record
```

```
origin: My3DPoint;
```

```
direction: My3DPoint;
```

```
end;
```

```
PolygonExtra = record
```

```
normal, rotX, rotY, center: My3DPoint;
```

```
radius2: float;
```

```
end;
```

```
PolygonExtraArray = array[0..0] of PolygonExtra;
```

```
StorageRecord = record
```

```
poly_extra: PolygonExtraArray;
```

```
{ must be at the end, since it's an extensible array }
```

```
end;
```

```
StorageRecordPtr = ^StorageRecord;
```

```
function DotProduct(const src1, src2 : My3DPoint) : float;
```

```
begin
```

```
DotProduct := src1.x3D*src2.x3D +
src1.y3D*src2.y3D +
src1.z3D*src2.z3D;
```

```
end;
```

### CrossProduct

```
procedure CrossProduct(src1, src2 : My3DPoint;
```

```
var dst : My3DPoint);
```

```
begin
```

```
dst.x3D := src1.y3D*src2.z3D - src1.z3D*src2.y3D;
```

```
dst.y3D := src1.z3D*src2.x3D - src1.x3D*src2.z3D;
```

```
dst.z3D := src1.x3D*src2.y3D - src1.y3D*src2.x3D;
```

```
end;
```

### AddVectors

```
procedure AddVectors(const src1, src2 : My3DPoint;
```

```
var dst : My3DPoint);
```

```
begin
```

```
dst.x3D := src1.x3D + src2.x3D;
```

```
dst.y3D := src1.y3D + src2.y3D;
```

```
dst.z3D := src1.z3D + src2.z3D;
```

```
end;
```



```

SubtractVectors
procedure SubtractVectors(const src1, src2 : My3DPoint;
                          var dst : My3DPoint);
begin
  dst.x3D := src1.x3D - src2.x3D;
  dst.y3D := src1.y3D - src2.y3D;
  dst.z3D := src1.z3D - src2.z3D;
end;

```

```

MidPoint
procedure MidPoint( const src1, src2 : My3DPoint;
                   var dst : My3DPoint);
begin
  dst.x3D := (src1.x3D + src2.x3D) / 2;
  dst.y3D := (src1.y3D + src2.y3D) / 2;
  dst.z3D := (src1.z3D + src2.z3D) / 2;
end;

```

```

Distance2
function Distance2( const src1, src2 : My3DPoint) : float;
begin
  Distance2 := sqr(src1.x3D - src2.x3D) +
               sqr(src1.y3D - src2.y3D) +
               sqr(src1.z3D - src2.z3D);
end;

```

```

ScaleVector
procedure ScaleVector(const src : My3DPoint; scale : float;
                     var dst : My3DPoint);
begin
  dst.x3D := src.x3D * scale;
  dst.y3D := src.y3D * scale;
  dst.z3D := src.z3D * scale;
end;

```

```

NormalizeVector
procedure NormalizeVector(const src : My3DPoint;
                          var dst : My3DPoint);
var
  length : float;
begin
  length := sqrt(DotProduct(src,src));
  dst.x3D := src.x3D / length;
  dst.y3D := src.y3D / length;
  dst.z3D := src.z3D / length;
end;

```

```

MakeViewRay
procedure MakeViewRay(const eye : My3DPoint;
                     x, y, z: float; var ray : Ray3D);
begin
  ray.origin.x3D := x;
  ray.origin.y3D := y;
  ray.origin.z3D := z;
  ray.direction.x3D := eye.x3D - x;
  ray.direction.y3D := eye.y3D - y;
  ray.direction.z3D := eye.z3D - z;
  NormalizeVector(ray.direction, ray.direction);
end;

```

```

RotateX
procedure RotateX(src : My3DPoint; sinA, cosA : float;
                 var dst : My3DPoint);
begin
  dst.x3D := src.x3D;
  dst.y3D := cosA*src.y3D - sinA*src.z3D;
  dst.z3D := sinA*src.y3D + cosA*src.z3D;
end;

```

```

RotateY
procedure RotateY( src : My3DPoint; sinA, cosA : float;
                  var dst : My3DPoint);
begin
  dst.x3D := cosA*src.x3D + sinA*src.z3D;
  dst.y3D := src.y3D;
  dst.z3D := -sinA*src.x3D + cosA*src.z3D;
end;

```

**OpenGL**® for the **Macintosh**™

The Industry Standard for 3D Graphics

- Port your code from other platforms with ease!
- Workstation Class Performance on your Mac
- Complete on-line documentation and support
- All common support libraries provided
- Multi-processor capable
- UNIX or MKLinux versions
- Works with most compilers, including CodeWarrior®
- Callable from C/ C++, Ada and FORTRAN
- Full Technical Support
- Custom Libraries available



**Conix Graphics**

Download free demos at:  
[www.conix3d.com](http://www.conix3d.com)

*Let your imagination fly!*

**Sales**

**800.577.5505**

**Technical**

**817.467.0461**

**FAX**

**817.467.9452**

**[sales@conix3d.com](mailto:sales@conix3d.com)**

All trademarks remain the property of their respective owners.

```

RotateZ
procedure RotateZ( src : My3DPoint; sinA, cosA : float;
                  var dst : My3DPoint);
begin
  dst.x3D := cosA*src.x3D - sinA*src.y3D;
  dst.y3D := sinA*src.x3D + cosA*src.y3D;
  dst.z3D := src.z3D;
end;

```

```

PointInPlaneInPolygon
function PointInPlaneInPolygon( const pt: My2DPoint; const
                                poly: MyPolygon ): boolean;
function Quadrant( const pt: My2DPoint; x, y: float ):
                                longint;
begin
  if pt.x2D > x then begin
    if pt.y2D > y then begin
      Quadrant := 0;
    end else begin
      Quadrant := 3;
    end;
  end else begin
    if pt.y2D > y then begin
      Quadrant := 1;
    end else begin
      Quadrant := 2;
    end;
  end;
end;

```



```

function x_intercept( const pt1, pt2: My2DPoint;
                      yy: float );
    float;
begin
    x_intercept := pt2.x2D -
        ((pt2.y2D - yy) *
         ((pt1.x2D - pt2.x2D)/(pt1.y2D - pt2.y2D)));
end;

var
    i, angle, quad, next_quad, delta: longint;
    last_vertex, next_vertex: My2DPoint;
begin
    angle := 0;
    last_vertex := poly.thePoint[poly.numPoints-1];
    quad := Quadrant( last_vertex, pt.x2D, pt.y2D );
    for i := 1 to poly.numPoints do begin
        next_vertex := poly.thePoint[i-1];
        next_quad := Quadrant( next_vertex, pt.x2D, pt.y2D );
        delta := next_quad - quad;
        case delta of
            3: delta := -1;
            -3: delta := 1;
            2, -2: begin
                if x_intercept( last_vertex, next_vertex, pt.y2D )
                    >
                    pt.x2D then begin
                    delta := -delta;
                end;
            end;
            otherwise begin
            end;
        end;
        angle := angle + delta;
        quad := next_quad;
        last_vertex := next_vertex;
    end;
    PointInPlaneInPolygon := (angle = 4) | (angle = -4);
end;

Intersect
function Intersect(const ray: Ray3D; const poly: MyPolygon;
                  const poly_extra: PolygonExtra; var distance: float;
                  var intersectionPt: My3DPoint): boolean;
var
    tempVector: My3DPoint;
    temp1, temp2: float;
    intersectionPoint: My3DPoint;
    intersection2D: My2DPoint;
    lb, lc, ld: float;
begin
    Intersect := false;

    { intersect ray with sphere }
    SubtractVectors( ray.origin, poly_extra.center,
                    tempVector );
    lb := 2 * DotProduct( ray.direction, tempVector );
    lc := DotProduct( tempVector, tempVector ) -
        poly_extra.radius2;
    ld := sqr(lb) - 4.0*Ic;
    if ld >= 0 then begin { yes, ray intersects sphere }
        temp1 := DotProduct( poly.polyPlane.planeOrigin,
                            poly_extra.normal ) -
            DotProduct( poly_extra.normal, ray.origin );
        temp2 := DotProduct(ray.direction, poly_extra.normal);
        if temp2 <> 0 then begin
            distance := temp1 / temp2;
            if distance > 0 then begin
                ScaleVector(ray.direction, distance,
                intersectionPoint);
                AddVectors(intersectionPoint, ray.origin,
                intersectionPoint);

                if Distance2(intersectionPoint, poly_extra.center)
                <=
                    poly_extra.radius2 then begin
                    { intersection point is within sphere.
                    Find out if it is actually in the polygon }
                    intersectionPt := intersectionPoint;
                    { First translate back to the origin }
                    SubtractVectors(intersectionPoint,
                    poly.polyPlane.planeOrigin, intersectionPoint);
                    intersection2D.x2D := DotProduct(
                        intersectionPoint,
                        poly_extra.rotX );
                    intersection2D.y2D := DotProduct(
                        intersectionPoint,
                        poly_extra.rotY );
                    { Then check if it is within the polygon }
                    Intersect := PointInPlaneInPolygon
                        (intersection2D, poly);
                end;
            end;
        end;
    end;
end;

InitProjection
procedure InitProjection(
    const viewPoint: My3DPoint; (* viewpoint from which to project *)
    const illumPoint: My3DPoint; (* viewpoint from which to draw shadow *)
    storage: univ Ptr; (* auxiliary storage preallocated for your use *)
    storageSize: longint (* number of bytes of storage *)
);
begin
    ($unused( viewPoint, illumPoint, storage, storageSize ))
end;

PreparsePolygons
procedure PreparsePolygons( my_storage: StorageRecordPtr;
    numPolys: longint; const thePolys: MyPolygonArray );
var
    i, j: longint;
    pt: My3DPoint;
    pts: array[1..kMAXPOINTS] of My3DPoint;
    min_x, min_y, min_z, max_x, max_y, max_z: My3DPoint;
    dist_x, dist_y, dist_z, new_radius2: float;
    radius, new_radius, old_to_new: float;
    sinX, cosX, sinY, cosY, sinZ, cosZ: float;
begin
    for i := 0 to numPolys-1 do begin
        with my_storage^.poly_extra[i], thePolys[i],
            polyPlane, planeNormal do begin
            sinX := sin(thetaX);
            cosX := cos(thetaX);
            sinY := sin(thetaY);
            cosY := cos(thetaY);
            sinZ := sin(thetaZ);
            cosZ := cos(thetaZ);
            normal.x3d := sinY*cosX;
            normal.y3d := -sinX;
            normal.z3d := cosY*cosX;
            rotX.x3D := 1;
            rotX.y3D := 0;
            rotX.z3D := 0;
            RotateZ(rotX, sinZ, cosZ, rotX);
            RotateX(rotX, sinX, cosX, rotX);
            RotateY(rotX, sinY, cosY, rotX);
            rotY.x3D := 0;
            rotY.y3D := 1;
            rotY.z3D := 0;
            RotateZ(rotY, sinZ, cosZ, rotY);
            RotateX(rotY, sinX, cosX, rotY);
            RotateY(rotY, sinY, cosY, rotY);

            for j := 1 to numPoints do begin
                pt.x3D := thePoint[j-1].x2D;
                pt.y3D := thePoint[j-1].y2D;
                pt.z3D := 0;
                RotateZ(pt, sinZ, cosZ, pt);
                RotateX(pt, sinX, cosX, pt);
                RotateY(pt, sinY, cosY, pt);
                pts[j] := pt;
                if j = 1 then begin
                    min_x := pt; min_y := pt; min_z := pt;
                    max_x := pt; max_y := pt; max_z := pt;
                end else begin
                    if pt.x3D < min_x.x3D then begin
                        min_x := pt;
                    end;
                    if pt.y3D < min_y.y3D then begin
                        min_y := pt;
                    end;
                end;
            end;
        end;
    end;
end;

```



# MACWORLD

E  
X  
P  
O

Unleash the **POWER**  
of your Macintosh

**W**ith so many products available to boost performance and creativity, your Mac has never had more potential than it does today. See the best and latest of these enhancements at MACWORLD Expo/Boston!

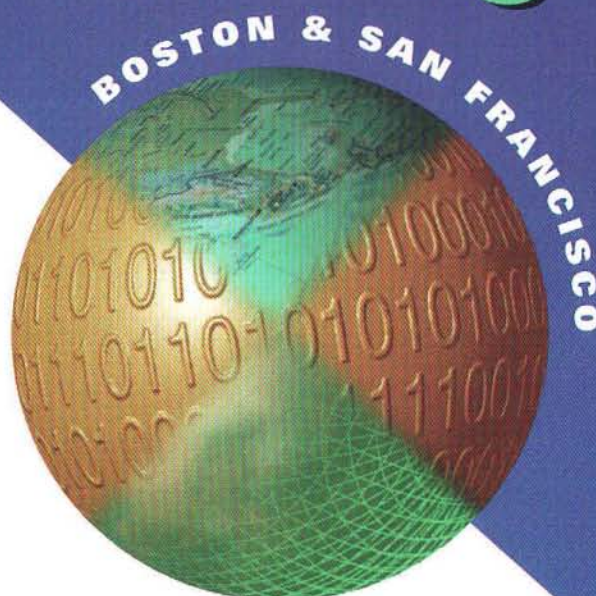
The product offerings at retail stores are no match for the breadth of choice you'll find at MACWORLD Expo. Catalogs may offer variety, but no opportunity to see demonstrations or ask questions. Only MACWORLD Expo lets you experience the full scope of Mac computing, with thousands of solutions for:

- Web site design and Internet navigation
- publishing, entertainment and multimedia
- networking, enterprise-wide connectivity, and intranets
- education, R&D, research
- business and telecommuting

The Mac universe is 60 million users strong. Rub shoulders with thousands of the most innovative, ingenious, and committed users at MACWORLD Expo. Network at special interest areas and pavilions that are free to all attendees. Learn from the experts... keep abreast of trends... maximize your Mac investment in more than 80 conference sessions. MACWORLD Expo opens your mind to new horizons accessible only through your Mac.

**INQUIRE TODAY  
FOR MORE  
INFORMATION  
ON MACWORLD  
EXPO.**

**Macworld  
EXPO**



**See us on the WWW at:  
[http://www.mha.com/  
macworldexpo/](http://www.mha.com/macworldexpo/)  
or call: 800-645-EXPO**

**Please send more information on MACWORLD Expo**

☐ Boston ☐ San Francisco

☐ Attending ☐ Exhibiting

MT

Name \_\_\_\_\_

Title \_\_\_\_\_

Company \_\_\_\_\_

Address \_\_\_\_\_

City/State/Zip \_\_\_\_\_

Phone \_\_\_\_\_ Fax \_\_\_\_\_

email \_\_\_\_\_

Mail to: MHA Event Management, 1400 Providence Highway,  
P.O. Box 9127, Norwood, MA 02062. Or Fax to: 617-440-0357

**THIS IS NOT A REGISTRATION FORM.**



```

    if pt.z3D < min_z.z3D then begin
        min_z := pt;
    end;
    if pt.x3D > max_x.x3D then begin
        max_x := pt;
    end;
    if pt.y3D > max_y.y3D then begin
        max_y := pt;
    end;
    if pt.z3D > max_z.z3D then begin
        max_z := pt;
    end;
end;

dist_x := Distance2( min_x, max_x );
dist_y := Distance2( min_y, max_y );
dist_z := Distance2( min_z, max_z );
if dist_x > dist_y then begin
    if dist_x > dist_z then begin
        radius2 := dist_x/4;
        MidPoint( min_x, max_x, center );
    end else begin
        radius2 := dist_z/4;
        MidPoint( min_z, max_z, center );
    end;
end else begin
    if dist_y > dist_z then begin
        radius2 := dist_y/4;
        MidPoint( min_y, max_y, center );
    end else begin
        radius2 := dist_z/4;
        MidPoint( min_z, max_z, center );
    end;
end;

for j := 1 to numPoints do begin
    new_radius2 := Distance2( center, pts[j] );
    if new_radius2 > radius2 then begin
        radius := sqrt(radius2);
        new_radius := sqrt(new_radius2);
        radius2 := (radius + new_radius)/2;
        old_to_new := radius2 - radius;
        center.x3D := (radius2*center.x3D +
            old_to_new*pts[j].x3D)/radius;
        center.y3D := (radius2*center.y3D +
            old_to_new*pts[j].y3D)/radius;
        center.z3D := (radius2*center.z3D +
            old_to_new*pts[j].z3D)/radius;
        radius2 := sqr(radius2);
    end;
end;

AddVectors( polyPlane.planeOrigin, center, center );

end;
end;
end;

```

#### CalcProjection

```

procedure CalcProjection(
    offScreen: GWorldPtr;           (* GWorld to draw projection *)
    const thePolys: MyPolygonArray; (* polygons to project *)
    numPolys: longint;              (* number of polygons to project *)
    const viewPoint: My3DPoint;     (* viewpoint from which to project *)
    const illumPoint: My3DPoint;    (* illumination point from which to draw shadow *)
    storage: univ Ptr;              (* auxiliary storage preallocated for your use *)
    storageSize: longint;            (* number of bytes of storage *)
);
var
    bounds: Rect;
    x, y: integer;
    colour: RGBColor;
    viewRay: Ray3D;
    lightRay: Ray3D;
    i: integer;
    closestDistance: float;
    closestIntersectionPt: My3DPoint;
    thisDistance: float;
    intersectionPt: My3DPoint;
    intersect_polygon: longint;

```

```

    pm: PixMapHandle;
    junk_boolean: boolean;
    pixels: Ptr;
    rowbytes_add: longint;
    my_storage: StorageRecordPtr;
begin
    ($unused( storage, storageSize ))
    my_storage := StorageRecordPtr(storage);

    PreparePolygons( my_storage, numPolys, thePolys );

    SetGWorld( offScreen, nil );
    bounds := offScreen^.PortRect;
    pm := GetGWorldPixMap( offScreen );
    junk_boolean := LockPixels( pm );
    pixels := GetPixBaseAddr( pm );
    rowbytes_add := band( pm^.rowBytes, $3FFF ) -
        4 * (bounds.right - bounds.left);

    for y := bounds.top to bounds.bottom-1 do begin
        for x := bounds.left to bounds.right-1 do begin
            MakeViewRay( viewPoint, x, y, 0, viewRay );
            closestDistance := 0.0;
            intersect_polygon := -1;
            for i:= 1 to numPolys do begin
                if Intersect( viewRay, thePolys[i-1],
                    my_storage^.poly_extra[i-1], thisDistance,
                    intersectionPt ) then begin
                    if (thisDistance > closestDistance) then begin
                        intersect_polygon := i;
                        closestDistance := thisDistance;
                        closestIntersectionPt := intersectionPt;
                    end;
                end;
            end;
            if intersect_polygon > 0 then begin
                colour := thePolys[intersect_polygon-1].polyColor;

                MakeViewRay( illumPoint, closestIntersectionPt.x3D,
                    closestIntersectionPt.y3D,
                    closestIntersectionPt.z3D, lightRay );

                for i:= 1 to numPolys do begin
                    if (intersect_polygon <> i) &
                        Intersect( lightRay, thePolys[i-1],
                            my_storage^.poly_extra[i-1],
                            thisDistance, intersectionPt ) then begin
                        colour.red := band(colour.red, $0FFFF) div 2;
                        colour.green := band(colour.green, $0FFFF) div 2;
                        colour.blue := band(colour.blue, $0FFFF) div 2;
                        leave;
                    end;
                end;

                LongintPtr(pixels)^ := bsl( band(colour.red, $0FFF0), 8 )
                    + band(colour.green, $0FFF0) +
                    bsr( band(colour.blue, $0FFF0), 8 );
            end else begin
                LongintPtr(pixels)^ := 0;
            end;
            longint(pixels) := longint(pixels) + 4;
        end;
        longint(pixels) := longint(pixels) + rowbytes_add;
    end;
end.

```

MT

Visit MacTech® Magazine's Web site!  
<http://www.mactech.com>



**If you want to be in the know, then you  
need every article published in the first  
12 years of MacTech® Magazine and  
Apple's develop™ issues 1-29!**

*...in THINK Reference™ format!*

**So hurry, pick up the phone, fire up  
the e-mail, launch that fax machine,  
or simply drop by our web site and  
order yourself this new release of  
the MacTech® CD-ROM.**

- Almost 1600 articles from all 139 issues of MacTech Magazine (1984-1996)
- Includes Apple develop issues 1-29
- Improved hypertext, improved indices, and a new THINK Reference Viewer — for lightning quick access!
- New hyperlinks between articles
- 100+ MB of source code — use them in your applications, with no royalties!
- Full version of THINK Reference — the original online guide to Inside Macintosh, Vols. I-VI
- 80MB of FrameWorks/SFA archives and the most complete set of FrameWorks archives known
- Sprocket™! MacTech's tiny framework that compiles quickly and supports System 7.5 features
- The best threads from the Mac programmer newsgroups plus thousands of notes, tips, snippets, and gotchas
- Popular tools that Mac programmers use to increase their productivity and much more!



**Developer  
DEPOT™**

Web Site: <http://www.devdepot.com> • E-mail: [orders@devdepot.com](mailto:orders@devdepot.com)  
Phone: 800-MACDEV-1 • Outside the U.S. & Canada: 805-494-9797 • Fax: 805-494-9798



by Nicholas C. "nick.c" DeMello, <URLs@mactech.com>

## Let the QuickTimes Roll

### QUICKTIME — MOVING PICTURES

"QuickTime, a set of functions and data structures that you can use in your application to control time-based data."

— from the *New Inside Macintosh: QuickTime*

Moving pictures impress. Your application can have a startling splash screen or a dynamic about box, and the QuickTime pages at Apple have almost 50 code examples to show you how. Of course, the essential reference for QuickTime is the *New Inside Macintosh: QuickTime* (ISBN 0-201-62201-7), written by Apple and published by Addison-Wesley's Developers Press. An electronic version of that book is available from Apples developers world pages in either Acrobat or Apple DocViewer format. Make sure to review the QuickTime tips at Terran Interactive and Charles Wiltgen's QuickTime FAQ before digging too deep into NIM.

Graphic Converter hasn't appeared at the info-mac archive since version 2.5, but the latest version (2.8) of this essential shareware utility can be downloaded directly from the Lemke Software pages. Among it's many features, Graphic Converter allows you to combine a series of static images into a single QuickTime movie. Localized versions in English, French, Swedish, Spanish, and German can be downloaded. Another valuable freeware tool is Movie Cleaner Lite, available from the Terran Interactive site. It supports conversion between Cinepak and Indeo codecs, for those of you moving QuickTime videos between MacOS and Windows.

#### Apple's QuickTime Pages

<<http://quicktime.apple.com/>>

#### QuickTime Sample Code

<<http://www.quicktime.apple.com/dev/devsw.html>>

#### Inside Macintosh: QuickTime

<<http://devworld.apple.com/dev/techsupport/insidemac/QuickTime/QuickTime-2.html>>

#### QuickTime Tips

<<http://terran-int.com/QTInfo/QTInfo.html>>

#### The QuickTime FAQ

<<http://www.QuickTimeFAQ.org/>>

#### Graphic Converter 2.7

<<http://www.lemkesoft.pingnet.de/>>

#### Movie Cleaner Pro

<<http://www.terran-int.com/prod/Cleaner.html>>

### QUICKTIME VR & QUICKDRAW 3D — MOVING INTO THE PICTURE

Just as tapes predated disks, linear access movies and sound tracks were only the first step in QuickTime. The QuickTime Media Layer now includes the new technologies QuickTime VR and QuickDraw 3D which allow users to interact with QuickTime movies

— to explore QuickTime worlds. QuickTime VR lets a user to rotate their perspective and experience a 360° view of a scene. It can also allow a user to roll an object in space — to see that same object from any side they choose. Apple has put a quick explanation of how QuickTime VR accomplishes both these tasks online, with more detailed information available at their QuickTime VR FAQ web page.

QuickTime VR can add an impressive degree of interactivity to your web pages and to your Director movies. The latest version of the QuickTime VR Plug-in for both environments is available from Apple. But Apple isn't stopping there. Current Apple plans include merging the QuickTime VR worlds with QuickDraw 3D technology. QuickDraw 3D allows you to define objects in space, then manipulate them within your code and dynamically render the scene into 2D form. Visit the QuickDraw 3D web pages to see dramatic examples of this 3D standard, and to download the latest version (1.5.1) of the QD3D extension. Nine develop articles have described how to implement QuickDraw 3D, and Apple has put them and their example code online at Developer World.

With QuickTime, worlds of sound and animation, panoramic vistas, and dynamic 3D objects can exist on your web sites and within programs. Let the QuickTimes roll...

#### How QTVR Works

<<http://quicktimevr.apple.com/how.html>>

#### The QTVR FAQ

<<http://quicktimevr.apple.com/dev/webfaq.html>>

#### QuickTime Plug-in 1.1 for Netscape

<<http://quicktimevr.apple.com/sw/qtvrmac.html>>

#### QuickTimeVR Plug-in for Director

<[http://quicktimevr.apple.com/bin/Forms/xtra\\_download.cgi](http://quicktimevr.apple.com/bin/Forms/xtra_download.cgi)>

#### QuickDraw 3D

<<http://quickdraw3d.apple.com/>>

#### develop QuickDraw3D Code Examples

<<http://devworld.apple.com/dev/techsupport/develop/bysubject/quickdraw3d.html>>

### PARTING SHOTS

Next month MacTech Online is going to get serious about Java. Until then, those of you who missed the 1996 WWDC might be interested in reviewing the slides from the imaging workshop, including Shawn Hopwood's overview of QuickDraw 3D (be sure to also download the player for the slides if you don't have Persuasion installed). See you next month.

— nick.c <<mailto:online@mactech.com>>

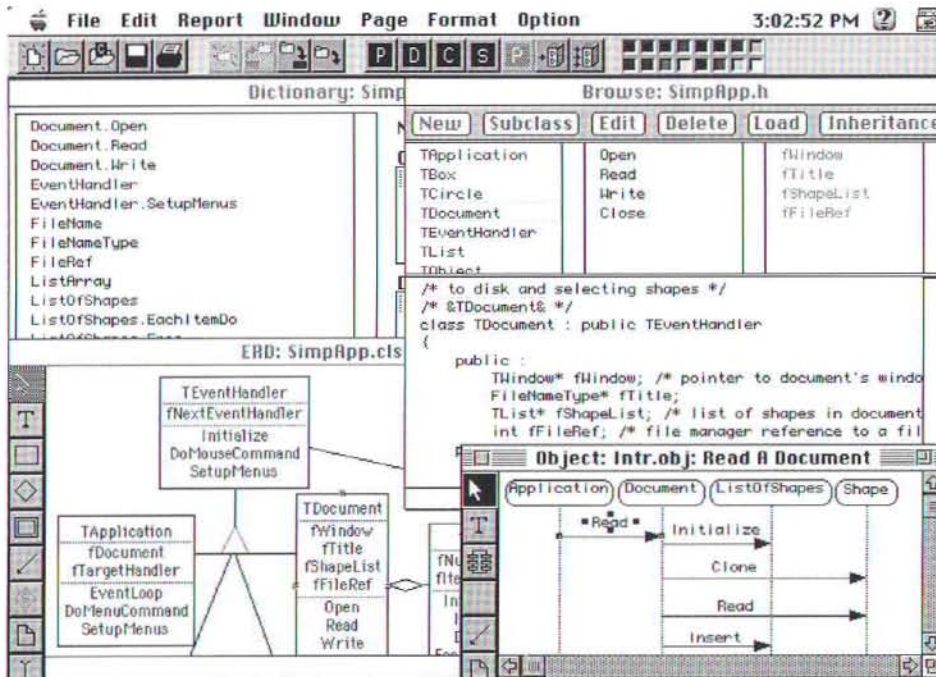
#### 1996 WWDC Overview of QuickDraw 3D

<[ftp://ftpdev.info.apple.com/Developer\\_Services/Programs/WWDC\\_Presentations/WWDC\\_1996/300\\_track\\_-\\_IMG\\_sessions/301\\_QuickDraw\\_3D\\_Overvi.sit.hqx](ftp://ftpdev.info.apple.com/Developer_Services/Programs/WWDC_Presentations/WWDC_1996/300_track_-_IMG_sessions/301_QuickDraw_3D_Overvi.sit.hqx)>

#### Persuasion Player for Slides

<[ftp://ftpdev.info.apple.com/Developer\\_Services/Programs/WWDC\\_Presentations/Persuasion\\_Player\\_3.0.sit.hqx](ftp://ftpdev.info.apple.com/Developer_Services/Programs/WWDC_Presentations/Persuasion_Player_3.0.sit.hqx)> 





The sporty new interface in MacA&D 6.0 enables designers to click to properties, dictionary details, integrated specifications, code or related requirements for any selected diagram object.

See <http://www.excelsoftware.com> for more details.

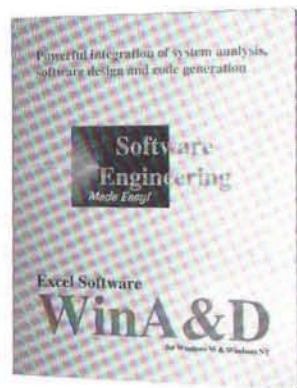
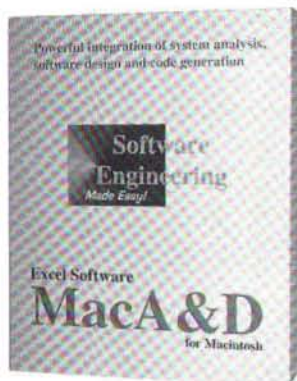
## Software Engineering Made Easy!

MacA&D and WinA&D simplify system analysis, track requirements specifications, automate popular modeling techniques and generate code from your design. Throughout the process, extensive verification reports catch errors early to help you produce a quality product.

Powerful capabilities are just one advantage. Years of innovation have made these products easy to use and each includes extensive examples and tutorials. MacA&D and WinA&D share documents so your project team can use Macintosh, Solaris, HP-UX or Windows 95/NT computers. Select from OMT, Booch, Shlaer/Mellor, Coad/Yourdon, Fusion or Jacobson methods or pick and mix the best of each.

- Structured Analysis & Design
- Object-Oriented Analysis & Design
- Data, State and Task Modeling
- Integrated Code Editing & Browsing
- Multi-User Dictionary
- Requirements/Use Cases
- Generates C, C++, Delphi, SQL, etc.

# Powerful Tools For Software Design



See our web site or drop us an email to receive detailed technical brochures for MacA&D or WinA&D. Check out the new MacTranslator and WinTranslator products to generate design documents automatically from existing source code.

**Excel Software**  
**515-752-5359**

**MacA&D**  
for Macintosh, Solaris and HP-UX

**WinA&D**  
for Windows 95/NT

© 1997 Excel Software. MacA&D, MacAnalyst, MacTranslator, WinA&D and WinTranslator are trademarks of Excel Software. Ph. 515-752-5359, Fax 515-752-2435, Email [info@excelsoftware.com](mailto:info@excelsoftware.com), Web: <http://www.excelsoftware.com>





by Andrew Stone, Chief Executive Haquer, Stone Design Corp.

# Porting to Rhapsody PPC from OPENSTEP

***Want your OPENSTEP application to run on more than one platform? Apple is making it easy***

In late 1995, when the future for NeXT Software, Inc. seemed rather bleak, the first beta version of OPENSTEP 4.0 shipped to a few faithful developers. Faced with doing Java development in incomplete environments, I decided to yet again recommit to OPENSTEP and port our best selling drawing and design program, *Create*, which contains the *Stone Libraries* upon which our other apps are based. In an effort to save others from the pitfalls I fell into, I produced a porting guide <http://www.stone.com/porting/> documenting the likely gotchas of the process of porting from NEXTSTEP to OPENSTEP.

It took me one month to port this 100,000 line code base to a state of linking and limping. After 8 months of intense debugging and working back and forth between the NT and MACH teams at NeXT and the SOLARIS OPENSTEP team at SUN Microsystems, *Create 4.0* rolled into beta, and was the first app to ship for OPENSTEP in the fall of 1996. Proving the claim of true cross platform compatibility, it ran on Windows 95 and Window NT, SOLARIS, and OPENSTEP MACH for Motorola, Intel and SPARC architectures. Since the development and porting tools were just being debugged for the first time and my OPENSTEP knowledge was still nascent, the process

was lengthier than a port from NEXTSTEP to OPENSTEP would be today. We shipped, and were underwhelmed by the lack of response from a beleaguered set of OPENSTEP users.

Apple acquiring NeXT on December 20th, 1996 changed all of that, proving once again that reality is far stranger than fiction. The Wheel turned full circle and the renegade ex-Mac developers were back home!

'Tis true that perseverance furthers, because 2 weeks prior to the 1997 WWDC on Cinco de Mayo, I was invited to NeXT Software's headquarters in Redwood City to build *Create* for the PowerPC. Unlike any previous visit to NeXT where I was watched with hawk-eye and not allowed to go anywhere unescorted, I was given full reign of the second floor of engineering, replete with my own office and access to that legendary espresso maker.

Two binaries were needed for the show: one for Rhapsody running on Intel, the other for Rhapsody running on PowerPC. The app was also built to showcase Yellow Box running on Windows with the Windows look and feel. This build compiled without warning, the app launched and hung. It turns out that I had a vestigial line of code from NEXTSTEP 1.0 circa 1989:

```
[menu display];
```

This was a workaround to get the menu to redraw after dynamically adding menu items. However, in the brave new world of Rhapsody, one cannot be sure what a menu is. Normally a menu is a subclass of NSView, which contains a display method. Under Rhapsody, Apple's drop-down menu is not a subclass of NSView, and does not recognize the display method.

The engineers at Apple/NeXT, in the midst of going through a total reorganization, had ported the kernel and compiler tools to PPC within a matter of months. They had built a cross-compiler to build PPC binaries from Intel machines, and once again, *Create* compiled cleanly. The PPC kernel was only a few days old, still dripping with amniotic fluid, when the first commercial app just came up and ran. As key team member Matt Watson is oft heard saying, "It just works!" Shouts of joy from the team could be heard throughout the building.

**Andrew Stone**, an early HyperTalk developer and co-author of "Tricks of the HyperTalk Masters" emigrated to the NeXT community in 1989, going on to write such NeXT classics as TextArt, Create, DataPhile and 3Dreality.

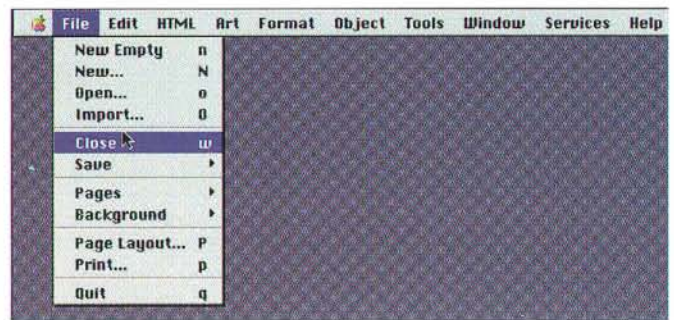




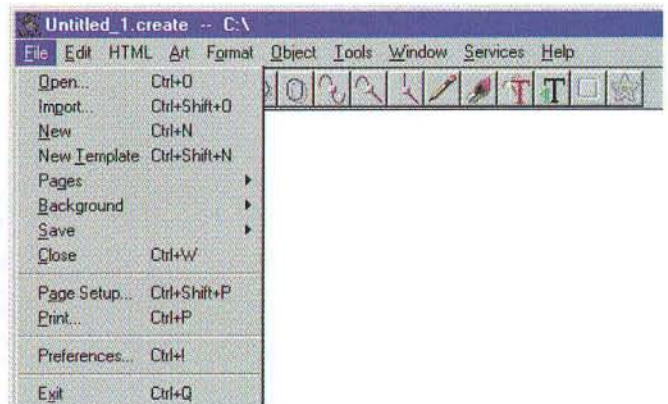
True to the promise of Rhapsody, no coding changes were required for the PPC/Yellow Box port, a few NeXT Interface Builder files (nibs) had to be adjusted to be more Mac-like — mainly *Create*'s main menu had to have its menu items reordered and renamed slightly to feel like a Mac. To solve this problem, a new interface loading paradigm was introduced: if a nib file has a platform suffix, that nib file is loaded instead of the one without a suffix:

Create		File
Info	⌘	New Empty n
File	⌘	New... N
Edit	⌘	Open... o
HTML	⌘	Import... O
Art	⌘	Pages
Format	⌘	Save
Object	⌘	Background
Tools	⌘	
Windows	⌘	
Services	⌘	
Print...	p	
Hide	h	
Quit	q	

*NEXTSTEP* look as provided by  
 \_NEXTSTEP\_\_<NIB\_FILE\_NAME>-nextstep.nib




*Macintosh* look as provided by  
 \_Macintosh\_\_<NIB\_FILE\_NAME>-macintosh.nib



*Windows* look as provided by \_Windows  
 NT/95\_<NIB\_FILE\_NAME>-windows.nib

The Mac and Windows NT/95 menus use a separation marker that you can find on the Menu palette of IB when you set the "Interface Style" in Interface Builder's Preferences (General) to "OPENSTEP for Windows." The separation marker must be disabled in the IB inspector to give it the light gray look. Menu organization and menu item naming should follow the User Interface guidelines of the target platform.

Having taken the time to learn Java and the various development environments from Sun and Symantec over the last couple of years, I can unreservedly say that Rhapsody rocks the house and Java has a lot to implement before it offers the power, elegance and integration of Rhapsody. After this delightful porting experience, I believe that **Rhapsody is Write Once — Runs Right Everywhere.** 





by Jessica Courtney

#### K Aidan Announces KiWi

##### A Low Cost Panoramic VR Tripod Head

KiWi is the first truly affordable panoramic VR tripod head. KiWi brings digital photographic panoramas to an even wider audience. With the advent of easy-to-use and affordable VR panorama creation software, such as PictureWorks' Spin and Panimation's Nodester, the KiWi panhead provides a complete solution that is ideal for anyone interested in adding digital VR panoramas to their websites, e-mail and multimedia applications.

The KiWi bracket consists of two intersecting aluminum struts that adjust and lock to accommodate the nodal points of a wide range of digital and conventional cameras, such as the Kodak DC50, Apple QuickTake 100/150/200, APS film cameras, up to 35mm SLR cameras equipped with 15mm lenses. When the two locking screws are loosened, the KiWi pan head disassembles into two flat pieces that easily fit inside a camera bag. Dimensionally, the adjustment range of the KiWi supports cameras with a tripod mounting thread-to-nodal point distance of 4 3/8 inches (111mm) and a camera mounting surface-to-nodal point distance of 5 1/4 inches (133mm).

KiWi attaches to any standard tripod and camera equipped with a standard 1/4-20 mounting thread. Unlike the other models of the Kaidan QuickPan line, which use angular detents to provide click-stops, the KiWi uses a specially designed friction joint that provides smooth and even drag while turning the camera. A convenient label on the base provides a clear indication of the location and increment angle of the camera. <info@kaidan.com>

#### QuickCRC 1.0 — Software Design Tool

Excel Software, makers of the popular MacA&D and WinA&D software engineering tools, today announced availability of QuickCRC 1.0 for both Macintosh and Windows 95/NT. QuickCRC is a software design tool for discovering objects and related information for an object-oriented software development project. QuickCRC automates the CRC card concept of identifying classes, responsibilities and collaborations between objects. It provides developers with a powerful, intuitive design tool that requires little or no formal training.

QuickCRC uses a diagram workspace for creating card and scenario objects. A card represents the properties of a class including its name, description, superclasses, subclasses, attributes, responsibilities and collaborating objects. A scenario represents a design mechanism defined as a series of steps involving communicating objects. Scenarios can reference cards or other scenarios. Information is entered into the Property dialog of each card and scenario object. Cards and scenarios can


also reference external agents defined by the designer to identify system and user interfaces for the software being designed.

QuickCRC provides active simulation of an evolving design. Features include single stepping backwards, forwards or over a called scenario or jumping to a specific location in the scenario stack of a multiple scenario simulation. Designers can even simulate mechanisms in an incomplete design by just selecting a scenario and clicking the Simulate button.

QuickCRC automatically and transparently maintains relationships between cards, scenarios and external agents as fluid design changes take place during the early phase of a development project. For example, if a card references undefined subclasses or superclasses, those cards are generated automatically. Name changes and cross references between objects get updated instantly. Card and scenario objects can be arranged visually on the diagram by QuickCRC based on user specified criteria to highlight relationships between objects. Card and scenario objects can be separated into different diagrams based on functional areas of a complex design or cut and pasted between different project documents. Design models can be verified to locate errors. Information can be listed to a text report to serve as a coding specification. Design models can also be exported or imported to Excel Software's MacA&D and WinA&D software engineering tools for detailed design or code generation. Likewise, Excel Software's MacTranslator and WinTranslator can be used to automatically generate QuickCRC design models from existing C++, Object Pascal or Delphi code. <http://www.excelsoftware.com>

#### Java News Offers Daily Distillate of Java

Roaster Technologies today officially launched "Roasted Java News," an online Java News Daily. Each freshly updated version includes a comprehensive collection of news headlines concerning Java culled from over 50 different publications available on the Internet. In addition, a department of "Short Takes" supplies information on new or updated applets, applications or Java-related sources.

Roaster Technologies' website also functions as a major Java resource for technology people working with Java. The newly revised site, also making its debut, includes a popular section called "Roasted Java Links," which provides an easily accessible wealth of information on Java related resources. This area gives the user current, verified information about other sites of interest to Java users; the user no longer has to wonder if the link is outdated, or stale. The webmasters at Roaster have done the work. <http://www.roaster.com/news/> 



# Developer Central<sup>TM</sup>

Sponsored by



*For Macintosh  
Programmers & Developers*

**MacTech<sup>®</sup>**  
M A G A Z I N E

---

**Macworld Expo/Boston • August 6 – 8, 1997**  
Bayside Expo Center

---

If you are developing Internet, multimedia, custom applications, or engineering solutions for the Mac<sup>®</sup> OS, Rhapsody and other Apple<sup>®</sup> platforms, then you need to visit Developer Central — your all-in-one source for the latest tools for Macintosh development.

Whether you want to jump-start your knowledge of

Macintosh development, or come up-to-speed on new tools or technologies, Developer Central will give you the inside scoop on Macintosh development by putting you in front of representatives from Apple Developer Relations and third-party tools developers. You'll even access the latest training and developer support programs offered by Apple.

## At Developer Central, you can...

- Talk directly to representatives from Apple and third-party vendors
- Demo developer products and tools
- Attend sessions in the Developer Central theaters
- Get the best deals on tools, training, and developer resource materials



So, no matter if you're developing a departmental system, using the Macintosh in science, developing plug-ins for the Internet, or planning the next great Mac OS application or multimedia title, *you'll find what you need at Developer Central.*

**Developer Central**  
**The Source for Macintosh Development**

---

**Sponsored by Apple Computer, Inc. and MacTech Magazine**





by Steve Sisak

In article <19970223210019110631@stcc010.ulaval.ca>, <brunet@geocities.com> (Charles Brunet) wrote:

When I call FSPDelete in my application, the document is not always deleted from the Finder. Sometime, I must create an other document with the same name to erase it. Is it possible to ask Finder to forget this file right now?

I have the same problem when I create a document. It take some time before it appear in the Finder.

How to update the informations in the Finder? If you have a scriptable finder, you can send it an update event. Here is a snippet from an (as of yet) unreleased piece of software:

```
OSErr
ICTyper::SendFinderAEOUpdate(FSSpec &inFile)
{
    OSErr err = noErr;
    AEDesc processDesc;
    AppleEvent ae, aeReply;
    ae.descriptorType = aeReply.descriptorType =
processDesc.descriptorType
= typeNull;
    ae.dataHandle = aeReply.dataHandle =
processDesc.dataHandle = nil;

    Try_ {
        DescType finderType = 'MACS';
        err =
::AECreatDesc(typeApplSignature,&finderType,sizeof(DescType),
&processDesc);
        FailOSErr(err);

        err = ::AECreatAppleEvent('fndr',
'fupd',&processDesc,
kAutoGenerateReturnID,kAnyTransactionID,&ae);
        FailOSErr(err);

        err =
::AEPutParamPtr(&ae,keyDirectObject,typeFSS,&inFile,sizeof(inF
ile));
        FailOSErr(err);

        err = ::AESend(&ae,&aeReply, kAENoReply |
kAENeverInteract,
kAENormalPriority,
kAEDefaultTimeout,nil,nil);
        FailOSErr(err);
    }
    Catch_(catchErr) {err = catchErr;} EndCatch_

    if (processDesc.descriptorType != typeNull)
::AEDisposeDesc(&processDesc);
    if (ae.descriptorType != typeNull) ::AEDisposeDesc(&ae);
    if (aeReply.descriptorType != typeNull)
::AEDisposeDesc(&aeReply);
    return err;
}
```

Jim Correia  
pmtb02jc@umassd.edu

### SCREEN SHOTS THE GNARLY WAY

One thing you can't do on the standard MacOS today is take a screen snapshot with a menu being displayed (or indeed, with

the mouse down). There are lots of third-party solutions to this, but in fact there is a way to get such a screen snapshot, by installing no additional software other than MacsBug. And since all readers of this magazine already have MacsBug installed (well, don't you?), that means you can indeed take screen snapshots with menus displayed on your Mac!

CAUTION: this technique can crash your machine, or have other unexpected consequences. Perhaps you should look on it as Another Fun Thing to Do With MacsBug, to show off to your fellow geeks at parties, rather than as a serious technique for regular use.

First of all, type cmd-shift-3 to invoke the standard screen snapshot FKEY. This is purely to make sure the FKEY code is loaded in memory; you can throw away the PICT file it creates at this point. Next, hold down the mouse button to display the menu that you want to take a snapshot of. While still holding the button down, use your other hand to press the interrupt button (or key sequence) to break into MacsBug. Once the MacsBug screen appears, you can release the mouse button.

Use the "hx" command, if necessary, to switch to the system heap. Now use the command

```
hd FKEY
```

to find out where the code for FKEY 3 is loaded. You will see a display that looks something like this:

```
Displaying the System heap at 00002000
Start Length Tag Mstr Ptr Lock Prg Type ID
File Name
0038F52C 0000046C+08 R 0019DA1C P FKEY 0003
0002
#1 block listed, which uses #1152 bytes, storing #1132
bytes
```

(If you see more than one FKEY loaded, the one you want will have "0003" under the ID column.) Make a note of the hexadecimal number under "Start" — that is the start address of the FKEY code. Next, enter the following commands:

```
sp := sp-4
sl sp pc
pc := xxxxxxxx
```

(replace the "xxxxxxx" with the start address of FKEY 3 you obtained before.)

Finally, enter the command "g" to resume execution. You should hear the click of the snapshot being taken, followed by the disappearance of the menu that was being displayed. But if you have a look at the resulting snapshot PICT file, you will see that it shows the menu!

Assuming your machine stays up that long (hee-hee)...

Lawrence D'Oliveiro  
<LDO@waikato.ac.nz>





**The Law Office of Bradley M. Sniderman**

California Lawyer focusing on Intellectual Property, Corporate, Commercial and Contract law, as well as Wills and Trusts.

If you are looking to protect your software with Copyright or Trademark protection, or if you need help establishing or maintaining your business, please give me a call or an e-mail. Reasonable fees.

**(310) 553-4054**  
**Brad@Sniderman.com**

**MacTech® Magazine  
is your  
recruitment vehicle**

When you need to fill important positions at your company, MacTech® Magazine is the consistent choice of companies across the country for hiring the best qualified Macintosh programmers and developers. Let MacTech® Magazine deliver your recruitment message to an audience of over 27,000 qualified computer professionals.

Call Ruth Subrin at 805/494-9797

**SOFTWARE ENGINEERS**

Lasergraphics is a dynamic computer graphics company located in Irvine, the center of Southern California's high-tech region. Our success is due to the innovation of our engineers; join us in developing code for high-speed manipulation of graphics, micro-optical control, self tuning systems, internet tools, and GUI apps. We are an evenly-mixed Mac/MS Windows environment.

For more details and contact info:

**<http://www.lasergraphics.com>**

Not online? Fax (714) 727-9282

EOE

No recruiters

**<http://www.scientific.com>**

**Professional software developers** looking for career opportunities should check out our web site. We offer nationwide employment assistance, resume help, marketability assessment, and never a fee to the applicant. 800-231-5920, Fax 800-757-9003 eMail: [das@scientific.com](mailto:das@scientific.com)

**Scientific  
Placement, Inc.**



**Wanna see your product *move*?**



**Get it in Developer Depot!**

**The fastest, cost effective way to get product off your shelves.**

For information: • Voice: 805/494-9797 • Fax: 805/494-9798 • E-mail: [vendor\\_relations@devdepot.com](mailto:vendor_relations@devdepot.com)



# ADVERTISER/ PRODUCT LIST

## LIST OF ADVERTISERS

<b>Aladdin Knowledge Systems Ltd.</b>	5
<b>Aladdin Systems, Inc.</b>	14-15
<b>Bare Bones Software, Inc.</b>	9
<b>BeeHive Technologies, Inc.</b>	29
<b>Bowers Development</b>	45
<b>Conix</b>	63
<b>Datapak Software Inc.</b>	7
<b>Developer Depot™</b>	95
<b>Excel Software</b>	69
<b>Faircom Corporation</b>	25
<b>Kaidan</b>	23
<b>Lasergraphics, Inc.</b>	75
<b>MacTech® CD ROM</b>	67
<b>MacTech® Now!</b>	73
<b>Macworld EXPO</b>	65
<b>Mango Tree Software, Inc.</b>	30
<b>Mathemaesthetics, Inc.</b>	1
<b>Metrowerks</b>	BC
<b>Micro Macro Technologies, Ltd.</b>	17
<b>MindVision Software</b>	48
<b>Miracle Software, Inc.</b>	47
<b>MotionsWorks International</b>	21
<b>Neologic Systems</b>	11
<b>Onyx Technology</b>	19
<b>Perforce Software</b>	41
<b>PrimeTime Freeware</b>	38
<b>Purity Software</b>	IBC
<b>Quasar Knowledge Systems</b>	10
<b>Ray Sauers Associates</b>	39
<b>Scapine Software, Inc.</b>	50
<b>Rivers Edge Corporation</b>	61
<b>Scientific Placement</b>	75
<b>Snowbound Software Corp.</b>	37
<b>Symantec</b>	IFC
<b>Water's Edge Software</b>	26

## LIST OF PRODUCTS

<b>ADB I/O • Beehive Technologies, Inc.</b>	29
<b>AppMaker • Bowers Development</b>	45
<b>BBEdit • Bare Bones Software, Inc.</b>	9
<b>CodeWarrior™ • Metrowerks</b>	BC
<b>c-tree™ • Faircom Corporation</b>	25
<b>Developer Tools • Developer Depot</b>	95
<b>DragInstall • Ray Sauers Associates</b>	39
<b>FairCom® Server • Purity Software</b>	IBC
<b>InstallerMaker • Aladdin Systems, Inc.</b>	14-15
<b>INStaller VISE • MindVision Software</b>	48
<b>MacA&amp;D • Excel Software</b>	69
<b>MacHASP • Aladdin Knowledge Systems Ltd.</b>	5
<b>MacRegistry™ • Scientific Placement</b>	75
<b>MicroGuard Plus™ • Micro Macro Technologies, Ltd.</b>	17
<b>MK Linux • PrimeTime Freeware</b>	38
<b>MotionWorks International • CameraMan</b>	21
<b>neoAccess™ • Neologic Systems</b>	11
<b>ODBC Driver • Faircom Corporation</b>	25
<b>OpenGL® • Conix Graphics</b>	63
<b>OpenPaige™ • DataPak Software, Inc.</b>	7
<b>Perforce • Perforce Software</b>	41
<b>RasterMaster 6.0 • Snowbound Software Corporation</b>	37
<b>Recruitment • Lasergraphics, Inc.</b>	75
<b>Recruitment • Scientific Placement</b>	75
<b>Resorcerer® 1.2 • Mathemaesthetics, Inc.</b>	1
<b>RiverReader-Laser • Rivers Edge Corporation</b>	61
<b>r-tree® • Faircom Corporation</b>	25
<b>SiteWeaver • Miracle Software, Inc.</b>	47
<b>SmallTalk Agents® • Quasar Knowledge Systems</b>	10
<b>Spotlight™ • Onyx Technology</b>	19
<b>TestTrack™ • Seapine Software, Inc.</b>	50
<b>TCP/IP Scripting Addition • Mango Tree Software</b>	30
<b>Tools Plus™ • Water's Edge Software</b>	26
<b>Trade Show • Macworld EXPO</b>	65
<b>Visual Cafe™ for Java • Symantec</b>	IFC
<b>WebSentinel™ • Purity Software</b>	IBC
<b>WebSiphon™ • Purity Software</b>	IBC

The index on this page is provided as a service to our readers. The publisher does not assume any liability for errors or omissions.



# Object Support Library Version History

*This article is an attempt to clarify the version history of Apple's Object Support Library (OSL). This library provides routines that applications can use to support the Open Scripting Architecture (OSA) object model.*

OSL was originally released as a 68K static library. With the introduction of Power Macintosh systems, OSL was repackaged as a shared library. When the Code Fragment Manager 68K Runtime Enabler (CFM-68K) was released, it became a fat library containing both Power PC and CFM-68K versions of OSL.

This article lists all versions of OSL that are currently available along with a brief history and description of each one, and recommendations as to which versions to use (and not use).

If you develop Macintosh applications that provide OSA object model support, or need to use applications that do, you should read this article.

## IN THE BEGINNING OF OSL

When the OSL was first released, it was a 68K static library (AObjectSupportLib.o) that was statically linked into applications that supported the OSA object model. This is still the case for classic 68K applications.

With the release of the Power Macintosh and the accompanying CFM shared library application model, it was decided that a shared library version of the OSL (ObjectSupportLib), rather than a '.o' file, should be provided for PowerPC-native applications. This would allow native applications to take advantage of the shared library application model.

The OSL PowerPC shared library was released as version 1.0.2. It was included with the first Power Macintosh system

software (version 7.1.2) and the AppleScript SDK version 1.1. It is currently still available on the Mac OS SDK CDs. This version of the OSL has a number of known bugs in its handling of whose clauses. (These bugs are listed in the Known Bugs in the OSL section of this Article.)

This first shared library introduced the first OSL shared library problem. The OSL was written in Pascal, but there were no PowerPC Pascal compilers, so it contained a small PowerPC-native library that loaded a 68K code resource containing the OSL.

But this loading was not done properly; the ObjectSupportLib's resource file often ended up in the middle of an application's resource chain. This problem was worked around in system 7.5.2 by forcing the Finder to load the OSL when it started up, thus forcing OSL's resource file to be located in the system, where it did no harm. This work-around masked the problem for all system versions after 7.5.2, but the problem is still there for earlier system versions.

After the introduction of the Power Macintosh, work began to convert the OSL source code from Pascal to C. This conversion was needed in order to provide a shared library containing both PowerPC and CFM-68K code. The converted fat OSL was released as version 1.0.4, which was included on the E.T.O. CD-ROM releases until recently.

There were a number of problems, however, with this version of the OSL, the worst of which was that a Gestalt selector installed by version 1.0.2 was not being installed by version 1.0.4. This prevented applications that were testing for this Gestalt selector from detecting the presence of the OSL shared library.

Note: A Gestalt selector is not the best method for detecting the presence of the OSL, but this decision was made early in the development of the code fragment model and the liabilities of using Gestalt were not yet well understood. The preferred method would have been to have developers compare a symbol in the library to `kUnresolvedCFragSymbolAddress`. (For details of this process, see Technote 1083: Weak-Linking to a Code Fragment Manager-based Shared Library)

The other major problem with this version of the OSL was a number of bugs in the code that handled whose clause resolution (in addition to those previously mentioned). These bugs would cause an application to return incorrect results and/or error messages when they were presented with valid requests. Because of the missing Gestalt selector which prevented this version's use, the whose clause resolution bugs were not discovered until much later.



## NO CODE CHANGES, JUST VERSION CHANGES

Somewhere between versions 1.0.2 and 1.0.4, a mistake was made in the build process for the AppleScript 1.1 SDK which led to version 1.0.2 of the OSL being released as version 1.1. There is no difference in the actual code, only a new version number.

This meant we now had a 1.1 version which is really older than version 1.0.4. But installers complain when you try to install v1.0.4 over 1.1 because they think you are replacing a newer version with an older one. So when Apple shipped the Apple Telecom software version of OSL, 1.0.4 was changed to 1.1.1. This allowed the installer to replace the old version with the new. Again, no code changes, just a version change. (All the old problems, of course, were still there).

This is the point where many people started having problems; we now had a situation where a (generally) well-behaved version 1.1 was being replaced with a broken version 1.1.1.

## A VALIANT ATTEMPT TO FIX THINGS

The AppleScript team, at this point, decided that enough was enough—it was time to move forward and fix things. They released a version that fixed the Gestalt selector problem and called it 1.1.1. But no one had told them about the Apple Telecom OSL version 1.1.1, so they changed the version to 1.1.2. But, again, someone else had already made a limited release of an OSL version 1.1.2.

This was unfortunate, but not a major problem—they just incremented the version to 1.1.3. This version, which contained the fixed Gestalt selector, got a limited developer release. Great—now the applications which couldn't load it before due to the Gestalt selector bug could use the OSL. But when these applications loaded the OSL, they revealed all sorts of interesting behaviors resulting from the OSL's inability to resolve whose clauses. (Remember that no one had actually executed this code before the Gestalt bug was fixed.)

## BACK TO THE DRAWING BOARD

The AppleScript team started looking at the 1.0.4/1.1.1 code stream in more detail and, after considering what a fix might mean to the stability of existing applications, it was decided that the risk of further trouble was too great. The decision was made to return to the older 1.0.2/1.1 code stream for the next release of the OSL.

As a result, the resource-loading source stream in version 1.0.2 was revised so it could be compiled as a native PowerPC and CFM-68K library. The resource-loading bug was also addressed. This revised library was tested, sent to a small group of developers for further testing, and then approved for final build as version 1.1.4.

Along the way, though, there was a new wrinkle added to this story. In order to fit in with the Mac OS reference release strategy, the file's creator type got changed, along with its version number, which changed to 1.1.6.

Note: Version 1.1.4 appears on the Developer CD Series and the Mac OS SDK CD, while version 1.1.6 appears on certain E.T.O. CDs.

But we still have more to our story. It seems that the test suite used to validate the OSL tested for the presence of the

Gestalt selector, but did not determine if it was properly installed. Basically, native OSL resource-loading code wasn't doing the right thing when it installed its Gestalt selector, which could lead to crashes in certain situations.

In the meantime, Version 1.1.6 was included as part of the Harmony (Mac OS 7.6) f3 build that was seeded to developers. When this latest OSL problem was discovered, the decision was made to take the version 1.0.2 of OSL and reversion it (again) as version 1.1.8. This is the version of OSL that is included in the GM version of Mac OS 7.6.

The biggest problem with this version is that it isn't fat, as it doesn't contain CFM-68K code (recall that 1.0.2 didn't, either). This wasn't an immediate problem, since Mac OS 7.6 shipped without support for CFM-68K. In fact, 7.6 explicitly checks for the presence of the current CFM-68K extension and disables it to prevent its loading with this version of the system. If you can't run CFM-68K, you don't need a fat OSL.

## A NEW HOPE

Thanks to the efforts of a number of engineers, the OSL got a more thorough rewrite which fixed both the Gestalt selector and resource file bugs. The testing was completed and the problems were resolved. This time, the OSL was released as version 1.2, which now generally works as expected. It still contains the bugs related to whose clause resolution, but they are not fatal and are easy for developers to work around (see Known Bugs in the OSL).

## THE WORLD AS WE KNOW IT

With all this said and done, you're probably scratching your head and wondering, "What version should I be using, anyway?" At this time (April, 1997) the answer to that question is (drum roll, please):

You should be using version 1.2 of the Object Support Library.

If you don't have access to this version, then use 1.1.8 (or one of its twins, versions 1.0.2 or 1.1). Keep in mind that these older versions are not fat, which means that they won't work with CFM-68K applications. If you need to run a CFM-68K application, you must use version 1.2.

And, of course, you should install newer version of the OSL as they are released.

## THE HISTORY OF OSL

Version	Status
1.0.2 .....	First PowerPC shared library, resource file bug
1.0.4 .....	New code base, no Gestalt selector, other bugs
1.1 .....	Really just 1.0.2 in disguise, new creator code
1.1.1 .....	1.0.4's twin (warts and all)
1.1.2/1.1.3 .....	Never publicly released, so don't look for it
1.1.4 .....	Doesn't work, but released on OS SDK CD
1.1.6 .....	Version that shipped with Harmony f3
1.1.8 .....	Version that shipped with Mac OS 7.6 (another 1.0.2 clone)
1.2 .....	Shipped with CFM-68K 4.0; fixes all known crashing bugs



## KNOWN BUGS IN THE OSL

There are still several known bugs in the OSL, but they all have workarounds.

### Unlocked Handles Passed To Compare Functions

You can install an object comparison function for use when resolving whose clauses. When the OSL calls your comparison function, it passes pointers to two AEDescs: one for the object being compared, and one for the object or descriptor to compare to the first.

The problem here is that the memory blocks containing the descriptor records pointed to by the parameters are located in relocatable blocks that are not locked and may move after the compare function is called.

The workaround for this problem is to copy the descriptor records pointed to by the parameters to a local variable as soon as you enter the object comparison routine.

There is a complication to this problem involving Classic 68K applications and the segment loader. If the comparison function and the AEOBjectSupportLib.o library are not in the same segment when you build your application, it's possible for the descriptor records to move before they are copied if the segment containing the OSL is not already loaded.

The workaround for this problem is to make sure that the comparison function and the AEOBjectSupportLib.o library are in the same segment when you build your application.

### Memory Leak in Object Comparison Callback Function During whose Clause Resolution

There is a memory leak when the OSL calls an application-supplied comparison callback function while resolving whose clauses in object specifiers. When the OSL passes objects to the comparison callback function, those objects will often be application-defined tokens that are created by object accessor functions.

The problem is that the OSL calls AEDisposeDesc on these token objects rather than AEDisposeToken, which causes a memory leak—any data the application has attached to the token is not properly disposed of.

This problem is further complicated by the unlocked handles bug described above. The workaround is to combine the unlocked handles fix with disposing the tokens yourself. The pseudo-code below describes the work-around for this problem and also the unlocked handle problem above:

#### Pascal version

```
function MyCompareObjects( comparisonOperator: DescType;
                          (CONST) VAR theObject: AEDesc;
                          (CONST) VAR objOrDescToCompare: AEDesc;
                          VAR compareResult: boolean): OSERR;
var
  theObjectCopy      : AEDesc;
  objOrDescToCompareCopy : AEDesc;
begin
  { First make copies of the descriptors because the OSL has them pointing
    into a relocatable block, which can be moved by the code below. }
  theObjectCopy := theObject;
  objOrDescToCompareCopy := objOrDescToCompare;
  { Now set the original descriptors to the null descriptor since we have
    to dispose of the objects below because of this OSL bug! }
  SetToNullDesc( theObject );
  SetToNullDesc( objOrDescToCompare );
  { Code to do the comparison goes here }
```

```
MyCompareObjects := DoTheComparison( comparisonOperator,
theObjectCopy,
```

```
objOrDescToCompareCopy, compareResult );
{ These descriptors are supposed to be const, but the OSL never calls
  our dispose token callback function, so we dispose of them here in
  case one of them is an application-defined token. }
MyDisposeToken( theObjectCopy );
MyDisposeToken( objOrDescToCompareCopy );
end;
```

#### C version

```
OSERR MyCompareObjects( DescType      comparisonOperator,
                        const AEDesc  *theObject,
                        const AEDesc  *objOrDescToCompare,
                        Boolean        *compareResult )
{
  AEDesc  theObjectCopy;
  AEDesc  objOrDescToCompareCopy;
  OSERR  anErr;
  theObjectCopy = theObject;
  objOrDescToCompareCopy = objOrDescToCompare;
  SetToNullDesc( theObject );
  SetToNullDesc( objOrDescToCompare );
  anErr = DoTheComparison( comparisonOperator,
                           theObjectCopy,
                           objOrDescToCompareCopy,
                           compareResult );
  MyDisposeToken( theObjectCopy );
  MyDisposeToken( objOrDescToCompareCopy );
}
```

### Memory Leak in marking Callback Functions During whose Clause Resolution

There is a memory leak when the OSL calls application-provided marking functions during the processing of whose clauses in object specifiers.

In the process of resolving an object specifier, a descriptor record is created and passed to the mark token callback function (as the containerToken parameter) and object-marking callback function (as the theToken parameter). The problem is that the OSL never disposes of this descriptor, either through AEDisposeToken or AEDisposeDesc.

The problem can be avoided by having your application resolve all whose clauses it receives on its own. This way it can avoid calling AEResolve. This can be advantageous, especially for the most common types of object specifiers. However, you can potentially receive "uncommon" object specifiers that are better handled by the OSL using your callback functions; if you handled them yourself you'd end up duplicating most of the OSL.

Fortunately, the workaround for this problem is relatively simple: dispose of the descriptor record at the end of the object-marking function and set it to a null descriptor. This fix should not cause problems if this bug is fixed in a future release of the OSL, since it's always safe to dispose of a null descriptor record.


### Function to set descriptor record to null descriptor

```
void SetToNullDesc( AEDesc theObject )
{
  theObject.descriptorType = typeNull;
  theObject.dataHandle = nil;
}
```

#### FURTHER REFERENCE

Technote 1083: Weak-Linking to a Code Fragment Manager-based Shared Library

#### ACKNOWLEDGMENTS

Special thanks to Chris Espinosa, Roger Pantos, Bryn Ob, and Otto Schlosser. 



# Newton Q&A: Ask the Llama

**Q:** I want to put my own custom icons in the picker that comes up when a user taps the action button. I have a slot in my base view called `myTestIcon` that contains the icon I want to use. But I can't figure out how to use that icon in the array of action frames in the `routeScripts` slot in my application. I started with the following routing item frame based on a simple "duplicate" action:

```
{title: "Test Action",
 icon: ROM_routeDuplicateIcon,
 routeScript: func(target, targetView)
 begin
 // my function here
 end,},
```

**This worked fine. Then I replaced the icon specification with this code:**

```
icon: GetRoot().(kAppSymbol).myTestIcon,
```

**But this won't even compile. Any ideas?**

**A:** The problem is that you're using the `GetRoot` function call in your array of routing action frames. Since the array of frames is in an evaluate slot (that is, the `routeScripts` slot), it will be evaluated at compile time in the Newton Toolkit environment. In other words, you're asking the Newton OS to look in the root view for your application's base view frame. This assumes that your application has been installed on a Newton device. But the code is executed at

compile time, so there is no Newton device, no root view, and no base application frame.

The easiest way to solve your problem is to define a constant for the icon in a text file and then use that constant in the appropriate slot that defines the action.

Let's assume that you already have a resource file with the appropriate PICT resource, and that you have a text file as part of your project (if not, just create one from the NTK File menu and add it to your project). In the text file, add a line that looks something like this:

```
DefineGlobalConstant('kMyIcon, GetPictAsBits("MyIcon",
nil));
// Note: Earlier versions of the platform file may use DefConst instead
// of DefineGlobalConstant.
```

This will give you a constant named `kMyIcon`. (Obviously you would replace `kMyIcon` with your own meaningful constant name and the string "MyIcon" with the name of the appropriate PICT resource.) Do this for each of the action menu icons that you want. Note that even though the action menu calls them "icons," you get the data by reading a Macintosh PICT resource, not a Macintosh ICON resource.

In your application's `routeScripts` slot where you create the array of extra action button actions, use the constant names you've created:

**The Llama** is the unofficial mascot of the Developer Technical Support group in Apple's Newton Systems Group. Send your Newton-related questions to [dr.llama@newton.apple.com](mailto:dr.llama@newton.apple.com). The first time we use a question from you, we'll send you a T-shirt.



```
{title: "Test Action",
 icon: kMyIcon,
 routeScript: func(target, targetView)
 begin
 // my function here
 end,1}.
```

The other way you can solve your problem is by specifying the array of action frames at run time. Instead of specifying a routeScripts slot with your array of action frames, you can provide a GetRouteScripts method. Since this method lets you construct your array of action frames at run time in the Newton environment, the root view will exist and your application will be found.

**Q: I'm using a protoLabelInputLine, which I've used many times before. In this case no user input happens in the input line. The entryFlags slot looks fine. It's set to vVisible, vClickable, vStrokesAllowed, vGesturesAllowed, and vAnythingAllowed. But strokes, gestures, and recognition do not occur. I tried adding a viewClickScript, viewStrokeScript, and viewGestureScript. The viewClickScript does get called, but none of the others do. When I remove the viewClickScript, there's still nothing. What's going on?**

**A:** You've fallen prey to a rare gotcha that's hard to find unless you know what you're looking for. The clue is in the behavior with and without the viewClickScript. Without it, the line takes no input, so it appears as if the view is ignoring clicks. With it, you see something occurring. The key is that the viewClickScript of a clickable view will be searched for in both the proto and the parent chain. So you must have a viewClickScript defined in a parent of the protoLabelInputLine. When the user puts the pen down in the input line, the inherited script is found and executed in the context of the input line. This inherited script probably turns off the ink and then does some processing. You've seen the result.

The solution is to add a viewClickScript to your protoLabelInputLine that returns the value 'skip. This will let the protoLabelInputLine grab the stroke and allow recognition to happen. Note that simply returning true or nil would not solve the problem.

**Q: The documentation for StuffLong says it "writes four bytes at the specified offset using the 30-bit signed value you pass it as a third parameter and sign-extends it to 32 bytes." Ignoring the obvious error ("32 bytes" should read "32 bits"), why does it take only 30 bits? I realize the language defines an integer as a 30-bit representation, plus a sign bit, but what's bit 31 for? More specifically, how can I use StuffLong to result in the value 0x20000000?**

**A:** First off, the representation is not 30 bits plus a sign bit plus some other bit. NewtonScript uses a 30-bit two's complement signed representation. The other two bits are used by the OS for other purposes. So StuffLong takes integers as 30 bits because that's how NewtonScript represents integers. In the binary object, the other two bits will be the sign extension of the number you pass in. That is, they'll have the same value as the sign bit: 0 for positive, 1 for negative.


As you can probably see, it isn't possible to use StuffLong to put your long word into a binary object. The two high-order bits must be the same as the sign of the number. In your case, you have a 1 bit in the sign, but you want a 0 for both bits of the sign extension.

To stuff the data you want, you'll have to use some combination of StuffWord and/or StuffByte.

**Q: I have some compile-time functions that generate frame data and some other functions that do sanity checking. Is it possible to abort the compile and give an error message once NTK has started compiling?**

**A:** Yes. NTK implements a subset of the Newton environment. This includes the ability to throw exceptions. To abort a compile, throw an exception. If the exception is of the type '!evt.ex.msg!, any message you provide will be shown in a dialog by NTK. Here's a simple example:

```
throw('!evt.ex.msg|, "Invalid data at position " &&
theIndex);
```

Assuming that theIndex has the value 4, this statement will abort the current compile and display a dialog containing "Invalid data at position 4." 





by Apple Developer Support Center

# Macintosh Q & A

**Q: FindSymbol returns a paramErr if I pass it a symbol name with a length greater than 64 characters, yet the documentation and interfaces specify that I pass the address of a Str255. What gives?**

**A:** This is a known bug, and has been fixed as of System 7.5.5. You should either make sure the System version is 7.5.5 or later, or keep your symbol name lengths to 64 or fewer characters.

**Q: Any time I try to load a shared library that has a nonzero implementation version number with GetSharedLibrary, CFM returns a fragTooNew error. Why?**

**A:** This is a bug in CFM that was fixed in System 7.5.5. If you need a nonzero implementation version number, you'll have to be sure to run only on System 7.5.5 or later. Otherwise, make your library's implementation version number 0.

**Q: I've been reading the documentation on the Resource Manager and I can't seem to find an unequivocal statement about whether resource IDs should be unique within a single resource type in a given resource file. Some documentation seems to suggest it's OK to have duplicate resource IDs as long as my program doesn't expect to be able to find resources by ID later. What's the real story?**

**A:** You're right; during a recent extensive survey of our documentation (*Inside Macintosh* old and new editions, Technotes, and so on), we found that some of the discussions of resource IDs were less than perfectly clear on this topic. The real story is that resource IDs should be unique within a

single resource type within a given resource file. When you're adding resources to a resource file, you might want to use UniqueID or UniqueIID to help you find unique resource IDs.

However, some resource files do contain multiple resources with the same type and ID. This can be due to bugs in the program that created the file or to the aforementioned ambiguity in the documentation. When reading resource files your program didn't create, make sure you can gracefully handle such files.

Regardless of what you may read in other documentation, the Resource Manager in some versions of the Mac OS may behave unpredictably when reading files that have multiple resources with the same type and ID.

One easy way to detect whether a given file contains multiple resources with the same type and ID is to open the file with Resorcerer or ResEdit. These programs will warn you if they detect this condition.

**Q: What's the logic behind the Create/Update preview behavior in the SFPGetFilePreview dialog? If I create previews for QuickTime movies I sometimes get preview movies and sometimes get preview pictures. With QuickTime movies that already have previews I sometimes get an Update button and sometimes get a dimmed Create button. What determines the behavior?**

**A:** PICT files (or files that QuickTime can import as PICT via the new graphic import components) don't have durations, so they can only have a preview PICT in the Standard File Preview dialog. Movies can have both a poster PICT and a movie preview.



When a movie or PICT file is selected in the dialog, the preview component will first see if a preview already exists in the file, stored in a 'pnot' resource. The 'pnot' resource also identifies whether the preview is a PICT, a movie, or whatever. The preview component then compares a timestamp in the 'pnot' resource to the modification date of the selected file to see if the preview is current. If the 'pnot' date is older than the last modified date of the file, the dialog will show the Update button (and will create the new preview with the 'pmak' components if the user selects this option).

If no 'pnot' resource is found in the selected file, and a 'pmak' component exists that can create a preview for the selected file type (QuickTime 2.5 supplies PICT, MOOV, and QTIF 'pmak' components), the Create button will be active in the dialog.

Note that sound files preview in a slightly different way. The 'pnot' components create an automatic 10-second preview (if there's that much sound) for supported sound file types without needing a 'pnot' resource in the file.

You can read more about preview components in *Inside Macintosh:QuickTime Components*.

**Q: I've saved files to disk in both foreground and background printing modes with LaserWriter 8.4.1, and the file size is always the same. I'd expect the size of the background image to be smaller as a result of compression and two-pass optimizations. Why don't I get different-size PostScript files?**

**A:** LaserWriter 8.4 (and all other 8.x versions) always saves to disk in two passes, hence the foreground and background files you see are the same size. If we didn't do this, we would generate non-DSC compliant PostScript jobs when saving to disk in the foreground. If you really want to check the PostScript code being sent to the printer, you should turn on the papToDisk bit in the driver's 'PRFS' resource. This creates papToDisk files that capture the data going to and coming from the printer.

**Q: I'm trying to find a way for my application to determine whether it's using a PostScript printer. For performance reasons, I'd like to send custom PostScript code instead of a PICT to the printer if I can. Is there an API to find out whether the currently selected printer is a PostScript printer?**

**A:** There's no good way of doing this. For Apple's LaserWriter drivers, you can, if you must, look at the wDev field in the print record of the currently selected printer. Here's a quote that's hidden in Technote QD 10, "Picture Comments — The Real Deal":

The high byte of the prStl.wDev field of the print record identifies a printer driver species; a value of \$03 tells you that the printer driver belongs to the PostScript LaserWriter driver ancestry...

However, although the Apple LaserWriter driver has a wDev of 3, other printer drivers for PostScript devices may not, so this is not a complete solution for detecting PostScript printers. Further, it's not recommended that you use the wDev field to distinguish between printer drivers at all; see the Print Hints column in this issue of *develop* for more on this.

**Q: I'm using the code from page 4-16 of *Inside Macintosh: Processes* to animate the cursor at VBL time, but it crashed (with the stack crawl indicating SetCursor as the culprit). What's up?**

**A:** When hardware cursor support was added to the Mac OS (System 7.5.2 for PCI Power Macs), SetCursor started requiring A5 to refer to a valid QuickDraw globals world. Despite the fact that the code you refer to accesses no global variables, your application still needs to make sure A5 is set up for SetCursor's benefit.

An explanation of setting up A5 in a VBL task can be found on page 4-13 of *Inside Macintosh: Processes*.

**Q: We're writing an application that requires us to connect to a remote machine via TCP/IP and talk to a background application running on that machine. However, we cannot connect to that machine when it's in sleep mode. Is there a way to keep the network services alive when a machine is in sleep mode? I've seen how you can keep the serial port alive, but not the network services.**

**A:** When a Macintosh (usually a PowerBook) goes into the "sleep" state, it's incapable of responding to network requests; the connections actually shut down. There are some Macintosh computers, however, that will attempt to go into an energy-efficient mode known as "doze."



The sleep state is easy to prevent and is pretty well documented in the Power Manager chapter of *Inside Macintosh: Devices* under "The Sleep Queue" and "Sleep Procedures," and there's more information in Technote 1046, "Inside Macintosh: Devices — Power Manager Addenda," and Technote 1086, "Power Management & The Energy Saver API."

If you want to prevent the system from sleeping or dozing, you should do the following:

1. Allocate a SleepQRec (preferably in the system heap).
2. Set it up to call into your sleep handler.
3. Return a nonzero value from your sleep handler when it's called. (When the Macintosh attempts to sleep or doze, your sleep handler is called with a sleepRequest or dozeRequest selector.)

In the doze state, Open Transport networking is still enabled and TCP connections that are set up should still function. But it might take several packets received within a short period (try 10 per second) to wake the machine from its doze state. You might also consider pinging the machine first to get it out of the doze state. Either way, if the machine is dozing it will take some time for the networking to reactivate, especially if virtual memory is enabled and the disk drive must spin up.

**Q: I'm writing a fax client-server system and I've encountered a problem after I put the server to sleep: on waking, the server software doesn't seem to reregister on the network (using RegisterMyName). On the client side, when receiving a sleep demand I wait until network activity has ceased, then return control to the system; the client wakes and reconnects to the server with no problems. What action should I take to correct the reregistering problem?**

**A:** In general, you might want to disable sleep on your server by informing the Power Manager with `AutoSleepControl(false)`. Otherwise, the clients might never know that your server is sleeping, and they'll be unable to connect. But if you do want to support sleep, a server should install a sleep procedure through the Power Manager using `SleepQInstall`.

The exact details of how your server should handle sleep requests and sleep demands are provided in Table 6-1 on page 6-10 of *Inside Macintosh: Devices*. Ultimately, you close and deregister your server from the network. Later, when you get the `sleepWakeUp` call, you should reopen and reregister.

**Q: I installed a sleep procedure, but the Power Manager will issue a sleep demand if the user selects Sleep from the Special menu. The `AutoSleepControl(false)` call will stop sleep requests but will it also stop sleep demands?**

**A:** Your application cannot refuse a sleep demand, as documented in *Inside Macintosh: Devices* on page 6-11:

When your sleep procedure receives a sleep demand, however, your procedure has no way to determine whether it originated as a conditional sleep demand or an unconditional sleep demand. Your device driver or application must prepare for the sleep state and return control promptly to the Power Manager when it receives a sleep demand.

For `AutoSleepControl`, see *Inside Macintosh: Devices*, page 6-44:

When `enableSleep` is set to false, the computer will not go into the sleep mode unless it is forced to either by some user action — for example, by the user's selecting Sleep from the Special menu of the Finder — or in a low battery situation.

**Q: How do I specify and control Open Transport serial port I/O handshaking? The options are a bit confusing.**

**A:** By using the `SRL_OPT_HANDSHAKE` option provided by the Open Transport native interfaces, you can customize serial port handshaking in a variety of ways. For instance, you can request that input handshaking be controlled by the CTS line or by the XON/OFF sequence. The default value of this option is no handshaking.

The handshaking behavior is specified by a 4-byte unsigned integer value that's passed in with the `SRL_OPT_HANDSHAKE` option. The high word (16 bits) of the integer is a bitmap with one or more of the following bits set:

<code>kOTSerialXOnOffInputHandshake</code>	= 1
<code>kOTSerialXOnOffOutputHandshake</code>	= 2
<code>kOTSerialCTSInputHandshake</code>	= 4
<code>kOTSerialDTROutputHandshake</code>	= 8

The second lowest byte is the XON character value, and the lowest byte is the XOFF character value. If these values are 0, and XON/XOFF handshaking was requested, the default values of Control-S for XOFF and Control-Q for XON will be used. An inline function (or a macro, for C users) is defined in `OpenTptSerial.h` for creating this 4-byte value:

```
OTSerialHandshakeData(UInt16 type, UInt8 onChar, UInt8 offChar)
```



Suppose, for instance, that you wanted to enable XON/XOFF input handshaking, but you wanted to specify that the XON character be Control-T instead of Control-Q. You would create an option structure as follows:

```
TOption    opt;
opt.len = kOTFourByteOptionSize;
opt.level = XTI_GENERIC;
opt.name = SERIAL_OPT_HANDSHAKE;
opt.value =
OTSerialHandshakeData(kOTSerialXOnOffInputHandshake,
    ('T' & ~0x40), // normally kOTSerialDefaultOnChar
    kOTSerialDefaultOffChar);
```

You can also control the XOFF state of the serial input port, by using the `I_SetSerialXOFFState` ioctl command. A value of 0 will unconditionally clear the XOFF state, while a value of 1 will unconditionally set it:

```
// Set XOFF state.
OIoctl(theSerialEndpoint, I_SetSerialXOffState, 1);
```

The `I_SetSerialXOn` ioctl command causes the serial port to send an XON character. A value of 0 will cause it to be sent only if we're in the XOFF state, while a value of 1 will unconditionally send the character:

```
// Unconditionally send an XON character.
OIoctl(theSerialEndpoint, I_SetSerialXOn, 1);
```

Conversely, the `I_SetSerialXOFF` ioctl command causes the serial port to send an XOFF character. A value of 0 will cause it to be sent only if we're in the XON state, while a value of 1 will unconditionally send the character:

```
// Unconditionally send an XOFF character.
OIoctl(theSerialEndpoint, I_SetSerialXOff, 1);
```

**Q: A friend told me you can make really great water rockets from plastic pop bottles. Is this true?**

**A:** Boy, howdy, is it ever! To give you an idea of how great they are, read the article "Soda Bottle Water Rockets" in *Physics Teacher* magazine, March 1995. Some interesting results from the article: If you launch a plain 2-liter bottle, filled with 0.7 liters of water (about 1/3 full turns out to be optimal) and pumped to 85 pounds per square inch, all the water is expelled in 0.07 seconds. At that point ("burnout") the bottle is only about 2 meters off the ground but is traveling at an incredible 76 meters per second (171 miles per hour)! That's an average acceleration of over 100 g's!

Of course, a naked bottle is aerodynamically hopeless, and tumbles fluffily after burnout. But if you add fins and some nose weight to make a stable rocket out of it, truly impressive altitudes are possible. A velocity-sensitive parachute deployment device has recently been concocted, bottle connection technology is advancing rapidly, and reliable multistage rockets are an active topic of investigation. Launching techniques range from the ultra-simple "cram a cork in it and pump 'til it blows" method to sophisticated, expensive launchers with all the bells and whistles.

For more information, see these Web sites: <http://members.aol.com/aquarocket/>, <http://www.H2ORocket.com/>, and <http://www.flash.net/~berggren/rocket/>.

**Q: If my Web server is running along happily under Open Transport (1.1 or 1.1.1), and the listener is bound to address 0.0.0.0, what happens when someone uses the control panel and changes the IP number? Right now it appears to just make the listener go deaf. I don't appear to receive connections on the new IP number, and if I use the control panel a second time to switch back to the original IP number, I don't get connections for that IP number either. Is there some event that gets sent to the listener that I'm not looking for that tells me when this happens?**


**A:** When a port changes its IP number, it's actually closing and reopening. When Open Transport closes a port, any endpoint that's plumbed to it is also closed, hence you'll get no further events on that endpoint.

The first thing you need to do is check for the provider events such as `kOTProviderWillClose` and `kOTProviderIsClosed`.

You should also use the `OTRegisterAsClient` call and register a notifier for client events, such as `kOTPortDisabled`, `kOTPortEnabled`, `kOTPortOffline`, `kOTPortOnline`, `kOTClosePortRequest`, `kOTYieldPortRequest`, and `kOTNewPortRegistered`.

When the interface changes, you need to close up your endpoints and rebind them.

---

*These answers are supplied by the technical gurus in Apple's Developer Support Center. For more answers, see the Macintosh Technical Q&As on the World Wide Web at <http://dev.info.apple.com/techqa/Main.html>. *



by Andy Bachorski and George Warner

## Mixed Up Threads

*See if you can solve this puzzle in the form of a dialog between a pseudo KON (Andy Bachorski) and BAL (George Warner). The dialog gives clues to help you. Keep guessing until you're done; your score is the number to the left of the clue that gave you the correct answer. Even if you never run into the particular problems being solved here, you'll learn some valuable debugging techniques that will help you solve your own programming conundrums. And you'll also learn interesting Macintosh trivia.*

KON So, BAL, I've run into a weird problem. I crash when I try to compile and execute a script using the AppleScript Open Scripting Architecture component from within a native PowerPC™ application.

BAL What's so weird about that? AppleScript hasn't been touched in years. It's bound to be showing its age by now.

KON Yeah, sure. Anyway, I've got an application that creates several threads.

BAL Now you're throwing the Thread Manager into the mix?

KON Look, can I just explain the problem? As I said, several threads are created, and inside each thread a string containing a valid AppleScript script is passed to the AppleScript component to be compiled and executed.

BAL This one's easy. You're using a single AppleScript component instance for all the threads. You need to have a separate instance for each thread for this scheme to work.

**Andy Bachorski** ([andyb@apple.com](mailto:andyb@apple.com)) works in Apple's Developer Technical Support answering questions about Apple events, AppleScript, and the Thread Manager. In previous lives he worked as a Mac consultant and system integrator, sold insurance door to door, was a used-car salesman, wore almost every hat while working at an Apple dealer, programmed CNC machining centers, walked the high iron as an iron worker, and wrenched on cars at Sears. Lifetime highlights include marrying his wife Linda and being present for the birth of their children Andrea and Colin.

**George Warner** ([geowar@apple.com](mailto:geowar@apple.com)) spends too much time searching for his wandering robot in the halls of Apple's R&D campus. He suspects the robot is jealous of his way-too-beautiful-to-marry-a-computer-nerd (but did) wife. Otherwise he's kept busy answering questions from developers about CFM and the Mixed Mode Manager in Developer Technical Support. Prior to his life at Apple, George shoved mainframes around at TRW and Digital and jumped out of perfectly good aircraft in the Air Force. Now he has a life (and a cute wife!).



100 KON Wrong. Why don't you let me finish? Inside each thread we ask for and get an AppleScript component instance. Next we call OSACompileExecute to tell AppleScript to compile the script text. The thread terminates when OSACompileExecute returns.

BAL Hmm. If all you're doing in each thread is calling OSACompileExecute, how are the other threads getting time before the first one terminates?

90 KON Well, the AppleScript component lets you install an active function that gets called periodically during the compilation and execution of a script. Normally you'd call WaitNextEvent to give time to other processes, but here we call YieldToAnyThread to give the other threads time.

BAL D'oh! I'll bet when you install the active function, you're passing a ProcPtr, and not a UniversalProcPtr.

80 KON I don't think so. I create a new routine descriptor by calling NewOSAActiveProc before installing the active function.

BAL Well, then, the routine descriptor is getting disposed of before the AppleScript component uses it.

KON Get a life.

BAL I've got it! The AppleScript component is 68K code, and since threads must be created and yielded from the same instruction set architecture, you crash when yielding from the active function.

70 KON No, I've installed a PowerPC routine as the active function, so I am creating and yielding the threads from the same ISA.

BAL Sounds as if the AppleScript component just doesn't like to be threaded.

60 KON No, that's not it, because I can run as many threads as I like with a 68K application. The crash happens only when the application is compiled as a native PowerPC application. I told you it was weird.

BAL I guess that means AppleScript and the Thread Manager aren't compatible on Power Macs.

50 KON Nope. As long as I create only a single thread in the native application it works just fine. But with two or more threads, I crash with a bus error.

BAL Maybe it would help if we knew where it's actually crashing. Set a breakpoint just before calling YieldToAnyThread in the active function.

45 KON After setting the breakpoint, I see that it's crashing when the second thread resumes after its first yield.

BAL That's a strange place to crash. Let's see what's going on inside the thread. I'm going to assume that you're opening a valid component instance.

KON Good call, since the thread would bail if it didn't.

BAL OK, so set a breakpoint on the call to OSACompileExecute.

40 KON Now we see that when the first thread resumes after its first yield, OSACompileExecute returns immediately with errOSAScriptError (-1753). If I continue and let the second thread resume, I crash with a bus error upon entering the second thread.

BAL But you say everything is fine if there's only a single thread. I'll bet the AppleScript component's A5 world is getting trashed. Why don't you save and restore A5 around the call to YieldToAnyThread?

35 KON After I add a call to SetCurrentA5 before yielding and a call to SetA5 after yielding, there's no change. Still crashing, same place, same way. Now what?

BAL Must be an internal flaw in AppleScript. It was written before either the Thread Manager or Power Macintosh systems existed. I don't think it's up to the task.



KON And I say you're wrong. But it sure seems like someone is getting confused whenever our active function is called.

BAL I thought you seemed a little dazed.

KON And confused!

BAL Let's see what the AppleScript component looks like before and after it calls the active function. Does the application still have the DebugStr calls in it?

KON Yes, but that won't do us any good. By the time we hit the breakpoint, the active function has already been called.

BAL Then it's time to get our hands dirty with MacsBug.

KON I'll go get the protective gear.

BAL Very funny. Let's start by seeing if the stack is getting munged by the call to the active function.

KON I'm yours to command.

BAL OK, run the application until the first thread is ready to return from the active function; then step until we're back in the AppleScript component.

30 KON We're in the component, and here's what you see when you disassemble around the PC:

00725C18	MOVE.L	A4, -(A7)	2F0C
00725C1A	JSR	*+\$1228 ; 00726E42	4EBA 1226
00725C1E	SUBQ.L	#\$2, A7	558F
00725C20	MOVE.L	D6, -(A7)	2F06
00725C22	JSR	(A3)	4E93
00725C24	*MOVE.W	(A7)+, D7	3E1F
00725C26	MOVE.L	A4, -(A7)	2F0C
00725C28	JSR	*+\$11BA ; 00726DE2	4EBA 11B8
00725C2C	EXT.L	D7	48C7
00725C2E	BEQ.S	*+\$0010 ; 00725C3E	670E

BAL The JSR to the address in A3 is where the active function is being called. We need to break just before it's called.

KON Great, I'll set a breakpoint at the JSR instruction.

BAL Not so fast. We've already returned from the function, so we'll need to restart the application to be able to check the stack before and after the active function is called.

KON And?

BAL And after we restart the application, the AppleScript component might not be loaded at the same place. We need the offset of this instruction in the component.

25 KON I'll use the thing dcmd to find out where the AppleScript component is loaded. After entering thing "osa " in MacsBug, we see this:

Displaying Registered Components

Cnt	tRef#	ThingName	Type	SubT	Manu	Flags	EntryPnt
#1	050009	(Not yet load...	osa	ascr	appl	000001FE	007059A0

BAL This gives us the location of the AppleScript component, and if we subtract the caller address from stack frame, we have the offset we need to find the instruction again.

KON OK, I've got its offset. I've restarted the application and, after locating the AppleScript component again, I've set a breakpoint before the active function is called.

BAL Go until you hit the breakpoint; then dump memory from the stack pointer before and after the JSR to the active function.

KON Except for the return value from the function, the stack looks untouched. Now what?

BAL So the stack is untouched. OK, let's restart and take a look at the registers.

KON Which ones?

BAL Let's start with the PowerPC registers and make sure the Thread Manager is doing the right thing.



20 KON We do a register dump at the breakpoints set in the active function before and after YieldToAnyFunction is called, and the registers look the same. The Thread Manager is doing the right thing.

BAL This time we'll check the emulated registers. Set the breakpoint at the active function call, and check the 68K registers around the call.

15 KON Restarted the application, set the breakpoint...we're there. Here's the registers display before the active function is called:

68020 Registers

D0 = 00000000	A0 = 0212E630
D1 = 00000005	A1 = 007275DE
D2 = 00000004	A2 = 022253E0
D3 = 00000001	A3 = 021674A0
D4 = 00000000	A4 = 0212E63C
D5 = 0000006B	A5 = 02265520
D6 = 00000000	A6 = 02156C9C
D7 = 00070000	A7 = 02156C82

And here it is after the active function returns:

68020 Registers

D0 = 00000100	A0 = 002E20D2
D1 = 0000200D	A1 = 00000017
D2 = FFFFFFF0	A2 = 0221C080
D3 = 00000001	A3 = 021674D0
D4 = 00000000	A4 = 0212E61C
D5 = 0000006B	A5 = 02265520
D6 = 00000000	A6 = 02156C9C
D7 = 00070000	A7 = 02156C86

It doesn't look good.

BAL Well, we can ignore D0 through D2, A0, and A1, because they're scratch registers and don't need to be saved. And we know that A6 and A7 will change as the application runs. That still leaves a lot of registers that aren't what they should be.

KON So who's twiddling our registers?

BAL Let's do the drill one more time, but this time we'll look at the registers around the call to the active function for both threads.

KON I've stepped through the calls to the active function, again.

BAL And?

10 KON When we return to the AppleScript component in the first thread, the emulated 68K registers contain the second thread's values.

BAL That would explain why AppleScript is crashing. It gets back after the call to the active function, but when it tries to pick up where it left off, someone has changed its world.

KON Now we know what's happening, but not why it's happening. We need someone to pin this on.

BAL So far we know that everything's fine in a 68K application. And we're OK with a single thread in a PowerPC application, but when there's a second thread we've got problems.

KON Right. And since we know that the emulated 68K registers are changing behind AppleScript's back, it must be the Mixed Mode Manager's fault.

BAL I'm not so sure. The Mixed Mode Manager is only responsible for transitioning between ISAs and RTAs (runtime architectures). It's up to the emulator to maintain the emulated registers, which are stored in a single emulator context block. I think this is causing problems for the Thread Manager.

KON What are you talking about? The Thread Manager doesn't have anything to do with the emulated 68K registers. Besides, the application works just fine when it's compiled as 68K code.

BAL It does, but when you have a 68K application, you have 68K threads and a 68K AppleScript component. In this environment, the registers all get properly protected.

5 KON The Thread Manager saves registers regardless of the ISA or RTA of the application.

BAL Right, but I think what we have here is a nasty interaction between the Thread Manager and the 68K Emulator. The root of the problem is the single emulator context block where the emulated 68K registers live.



KON Let's see. Inside the thread the AppleScript component (68K) calls the active function (PPC code) which causes a mode switch. Now we're in native mode with the emulated registers lying dormant in the emulator context block. At this point the AppleScript component is assuming that the active function it just called will preserve any nonvolatile registers it touches.

BAL Next the active function calls YieldToAnyThread; the Thread Manager saves a PowerPC thread context and goes to the next thread. But the Thread Manager doesn't know anything about the AppleScript component needing the emulated 68K registers preserved during the thread switch.

KON So now the second thread gets control, and it fills the emulated 68K registers with its own values. Another yield happens, and we get back to the first thread. But the single emulator context block now contains register values from the previous thread.

BAL And since that's not what the first thread is expecting, it dies a terrible death. Let's throw together a little hack that we can use to test your theory. First create a couple of inline 68K routines that save and restore the registers.

```
static UInt16 SaveRegisterHack[] = {
    0x48d0, 0x1cf8,    // movem.l d3-d7/a2-a4,(a0)
    0x4e75             // rts
};
```

```
static UInt16 RestoreRegisterHack[] = {
    0x4cd0, 0x1cf8,    // movem.l (a0),d3-d7/a2-a4
    0x4e75             // rts
};
```

KON That's really skanky. I like it!

BAL Now we need some procedure information, so we can call these routines.

```
enum {
    uppRegisterHackProcInfo = kRegisterBased
    | REGISTER_ROUTINE_PARAMETER(1, kRegisterA0,
    SIZE_CODE(sizeof(Ptr)))
};
```

KON Check.

BAL Now in the active function add a call to SaveRegisterHack before calling YieldToAnyThread and a call to RestoreRegisterHack after the yield call.

```
long reg68K[8];
CallUniversalProc((UniversalProcPtr) &SaveRegisterHack,
    uppRegisterHackProcInfo, reg68K);
YieldToAnyThread();
CallUniversalProc((UniversalProcPtr) &RestoreRegisterHack,
    uppRegisterHackProcInfo, reg68K);
```

KON Don't we need to create a UniversalProcPtr before calling the register hack routines?

BAL No. Remember, a 68K ProcPtr is a valid routine descriptor. Just call that sucker!

KON Cool!

BAL Here's where the rubber meets the road. Let's run the application now that we've got the registers protected.

KON Bingo. It ran all the way through without a crash. That proves our theory. You know, the problem is really that the Thread Manager was designed to function in a homogenous application context, but in a case like this we're abusing it with cross-ISA/RTA callback functions it doesn't expect. So it needs our help to keep everything straight.

BAL Too bad the hack we put together is so ugly. I'd never want to use something like that in my code.


KON You don't have to! Through special arrangements, BalKon Industries is able to offer on this issue's CD not one, but two, library routines you can link into your application to work around the active function problem, as well as a similar problem when installing send functions in a native application using the AppleScript component.

BAL Nasty.

KON Yeah.

#### SCORING

- 80-100 And I'll bet your code compiles and runs the first try.
- 45-70 You're an asset to your company; ask for a raise.
- 25-40 Not too bad, but keep your resumé updated just in case.
- 5-20 At least you're honest.

*Thanks to Eric Anderson, Bo3B Johnson, Jon Pugh, Quinn "The Eskimo!", KON (Konstantin Othmer), and BAL (Bruce Leak) for reviewing this column. *



# Developer **DEPOT**™

- Web site: <http://www.devdepot.com> • E-mail: [orders@devdepot.com](mailto:orders@devdepot.com)
- Phone: 1-800-MACDEV-1 (Outside the U.S. & Canada: 805-494-9797)
- Fax: 805-494-9798

- ✓ **Hundreds of developer products**
- ✓ **Satisfaction and lowest price guaranteed**
- ✓ **World renowned customer service**
- ✓ **Order by phone, E-mail, fax or through our continually updated Web site**



**NOW CARRYING  
APPLE PRODUCTS!**



***New Categories!***

- **Multimedia**
- **Games**



# We've got what you want!

Better still, we have the products you *need*  
with the prices and service you *deserve!*



**Developer Depot™** has been in the Macintosh Development tools business since 1984 and is committed to supporting the MacOS developer community. We offer all the usual great stuff, plus a wide variety of awesome new products — *all at the lowest prices guaranteed!*

**Developer DEPOT™**  
<http://www.devdepot.com>

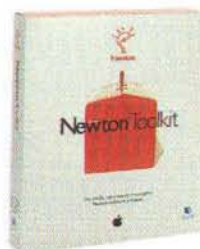
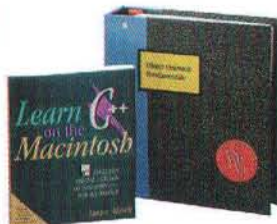
Tel: 800/MACDEV-1 • Outside U.S./Canada: 805/494-9797 • Fax: 805/494-9798  
E-mail: [orders@devdepot.com](mailto:orders@devdepot.com) • Web Site: <http://www.devdepot.com>



# ANNOUNCEMENT

## Developer DEPOT™

now carries Apple  
brand products



### Developer Depot 30 day Money Back, Price and Satisfaction Guarantee

Developer Depot products are sold with a 30 Day money back guarantee on user satisfaction, lower prices and against defects. If, for any reason, you are not satisfied or find the same product at a lower price within 30 days, please call Customer Service at 800-MACDEV-1 and request a Return Merchandise Authorization (RMA) number to get a full refund or the difference in price (where applicable). You must return undamaged product at your expense, including all its original packaging, documentation and the blank warranty

Stuff our lawyer made us write.

Developer Depot makes no other warranties. All other warranties, either expressed or implied, including the implied warranty of merchantability and fitness for a particular purpose are disclaimed. Developer Depot shall not be liable for any direct, special, incidental or consequential damages including lost profits, from any delay in delivery, or for any personal injury arising from the use of any product sold through Developer Depot. The limit of direct damages, if any, shall not exceed the purchase price of the product. © 1997 Xplain Corporation. All rights reserved. Any unauthorized duplication is in violation of federal laws. Developer Depot is a registered trademark of Xplain Corporation. All product names in this catalog are the trademarks of their respective holders.

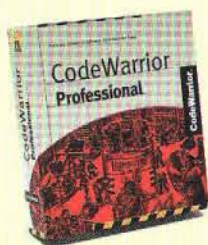
© 1997 Xplain Corporation. All rights reserved. Any unauthorized duplication is in violation of federal laws. Developer Depot is a trademark of Xplain Corporation. All product names in this catalogue are trademarks or registered trademarks of their respective holders

card if applicable. Developer Depot will replace defective product upon receipt of the defective merchandise. Please remember to back up your data before installation of any new hardware, software, or peripherals; we cannot be responsible for any lost data. Policies, item availability, and prices are subject to change without notice. The price in effect when we receive your order will be the price that you are charged. We are not responsible for any typographical errors in this or any other catalog, nor for any misstatements from any vendor. Purchase orders are not accepted without prior approval. Call for more information.

02	DEVELOPMENT ENVIRONMENTS
06	TOOLS, LIBS & UTILITIES
12	INTERNET RELATED
14	SCRIPTING
15	MULTIMEDIA
17	GAMES
18	TRAINING
19	ACCESSORIES
20	BOOKS & REFERENCE
31	INDEX







## CodeWarrior Professional Release 1 by Metrowerks

More platforms, more languages, more options: CodeWarrior Professional is designed to give you the tools you need for serious, industrial-strength programming. CodeWarrior Professional is the only Integrated Development Environment (IDE) in which you can edit, compile and debug C, C++, Pascal and Java programs for multiple target processors and operating systems. CodeWarrior's compilers produce fast, highly optimized code for Windows 95/NT running on x86, or Mac OS running on 68K or PowerPC processors. CodeWarrior Professional features the new CodeWarrior IDE Version 2. The newly revamped Project Manager supports multiple

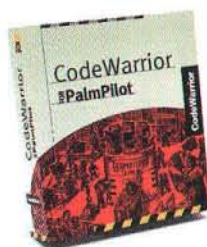
projects open simultaneously, multiple targets per project, and threaded execution, plus it's significantly faster. CodeWarrior Professional also includes online books, documentation, and reference materials, as well as tutorials and sample code. We support your development efforts with one free update and free world-class technical support for a year with registration.

- Includes Windows 95/NT and Mac OS versions of the CodeWarrior IDE
- Supports C, C++, Pascal and Java
- Develop for Windows 95/NT on x86 and Mac OS on 68K/PowerPC from either version
- New Project Manager supports multiple open projects, subprojects, multiple targets per project, and threaded execution
- Integration of tools means you spend less time navigating and more time coding

(SCWPRO) Our Price **\$599**

Upgrade for competitive product owners

(SCWPRUP) Our Price **\$449**



## CodeWarrior for PalmPilot by Metrowerks

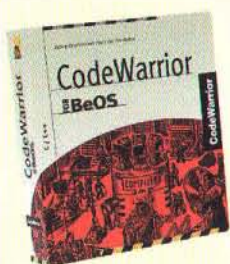
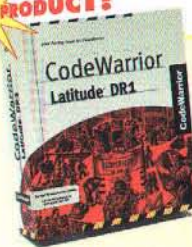
CodeWarrior for PalmPilot is the first complete set of tools which enables you to develop for the U.S. Robotics PalmPilot connected organizer, from the comfort of your PC or Macintosh computer. All the tools you need are included: the award-winning CodeWarrior IDE, compiler, linker, source-level debugger, GUI builder, and other tools, plus online documentation and reference materials. Includes one year of product updates and free technical support.

(SCWPALM) Our Price **\$369**

## CodeWarrior Latitude by Metrowerks

Don't throw away the investment you have made in your Mac OS applications! With the DR/1 release of CodeWarrior Latitude, you can prepare your Macintosh applications to run native under Silicon Graphics IRIX or Sun Solaris (and soon, Rhapsody). Begin work now to prepare your Mac OS application to run native under Rhapsody. By compiling and linking your CodeWarrior project with the Latitude libraries, you can identify which portions of your code will port smoothly and which Mac Toolbox traps are not implemented. Additionally, you can expand the market for your Mac OS application by completely porting it to Silicon Graphics IRIX or Sun Solaris at a fraction of the cost of other porting tools. As the Rhapsody API evolves, so will Latitude. Registered users of CodeWarrior Latitude will receive all developer releases, the first full release, plus one additional product update, to ensure that you have access to the most up-to-date Rhapsody porting tools available.

(SCWLAT) Our Price **\$399**



## CodeWarrior for BeOS 3 by Metrowerks



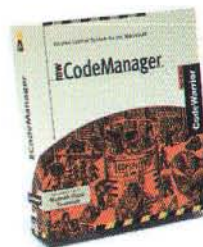
- Start programming for the new, innovative Be Operating System (BeOS) with complete set of Codewarrior tools
- BeOS-native Integrated Development Environment (IDE) with all the familiar CodeWarrior features at your fingertips
- A BeOS PowerPC compiler and linker, an editor w/syntax color and styling, and a source-level debugger
- BeOS header and libraries, complete documentation, useful C++ classes, and sample code
- The Be Operating System Advanced Access Preview Release allows you to run and program for the BeOS on a 603 or 604 PCI based PowerMac

(SCWFB) Our Price **\$149**

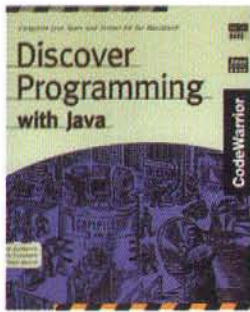
## CodeManager 4 by Metrowerks

- Source code control system, plug-in to the CodeWarrior IDE
- Compatible with Microsoft Visual SourceSafe version 5.0
- Cross-platform support (Mac, Windows, UNIX, OS/2)
- Configuration management in excess of 4 billion files
- Over 8,000 files and sub-projects in a single sub-project
- Registered users receive one year of free technical support and two free updates

(SCDMGR) Our Price **\$399**







## Discover Programming with Java

by Metrowerks

Discover Programming with Java is a professional Java environment, providing the most complete toolset available on the Macintosh. All your projects can be built from the ground up with the award-winning CodeWarrior Integrated Development

Environment (IDE), which boasts a full-featured editor, a Sun-validated Java compiler, a source-level debugger and a Java class browser. Preview your Java programs using the applet viewer, which is conveniently linked to the debugger for easy testing.

- Full Java toolkit: project manager, linker, editor, class browser, source-level debugger, applet viewer and more
- Includes an online book, plus online documentation, reference materials and tutorials
- One year free technical support with registration (SCWDISCJAVA) Our Price **\$79**



## Newton Toolkit 1.6

by Apple Computer, Inc.

With Newton Toolkit, you can easily create software that runs on any Newton PDA, including Apple's MessagePad and Motorola's Marco.

- Now supports Newton 2.0
- Dynamic Language Eases Development
- Allows you develop applications interactively
- New Compiler Enables Faster Applications

Newton Toolkit 1.6

(SNETO) Our Price **\$149**

Newton Toolkit Update 1.6

(SNETOUP) Our Price **\$29**



## Apple Dylan

dynamic object oriented language and development environment

## Apple Dylan Technology Release

by Apple Computer, Inc.

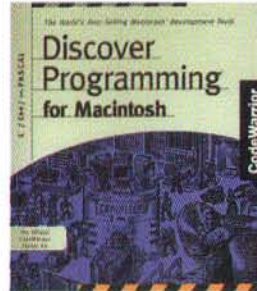
- Contains a PowerPC-native prototype version of a development environment based on the Object Oriented Dynamic Language (OODL) Dylan. Developers will be able to produce code targeting both 680x0 and Power Macintosh systems
- Automatic memory management
- Application framework and user-interface & builder
- High-level exception handling
- Cross-language support for C code and APIs

CD & Online Documentation

(SADTRO) Our Price **\$39.95**

CD & Hardcopy

(SADTRH) Our Price **\$59.95**



## Discover Programming for Macintosh

by Metrowerks

Discover Programming for Macintosh provides everything you need to start and complete your programming education. Full-featured, award-winning CodeWarrior tools allow you to build full-blown 68K Macintosh applications. Powerful reference material and online

books, including Learn C on the Macintosh, Learn C++ on the Macintosh by Dave Mark, and Jim Trudeau's Programming Starter Kit for Macintosh are included on the CD-ROM. Plus, the book examples have been converted to Apple Guide files for help that is just a mouse-click away. Loads of source code is also included to help get you started on the road to success.

- CodeWarrior Integrated Development Environment (IDE) with C, C++, and Object Pascal compilers for 68K Mac OS
- Includes four online books, plus online documentation, reference materials and tutorials
- One year free technical support with registration (SCWDISCMAC) Our Price **\$79**

## Newton Toolkit 1.6 for Windows

by Apple Computer, Inc.

With Newton Toolkit, you can easily create software that runs on any Newton OS device from Apple or its licensees. The Newton Toolkit is a development environment that reduces the complexity and time involved in creating great software products.

- Provides access to features specific to the Newton 2.0 OS - Allows developers to create new or modify existing Newton applications to take advantage of powerful Newton 2.0 OS features
- Supports Windows 95, Windows NT 3.5x, and Windows 3.1(with Win32s) operating systems -Enables developers to use multiple development environments/operating systems when creating Newton applications
- Includes Macintosh-to-Windows NTK project converter tool - Allows developers to automate conversion of existing NTK projects from Macintosh platform to Windows platform
- Provides NS Debug Tools - Allows developers access to interactive Newton application debugging tools. Offers extended NewtonScript debugging functions, which let developers study and manipulate an application running on a Newton device. Developers can use NS Debug Tools to set break points in an application after it's been compiled and installed, step through program execution, examine and change objects on the Newton, and display a textual representation of the interpreter instructions

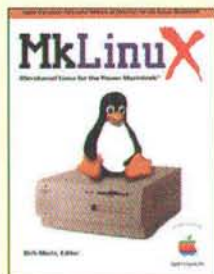
(SNETOW) Our Price **\$149**

Newton Toolkit 1.6 Update

(SNETOWUP) Our Price **\$29**







**MkLinux: Microkernel  
Linux  
for the Power Macintosh**  
by Prime Time Freeware



MkLinux is a native port of Mach 3 and the Linux 2.0 kernel, complemented by hundreds of commands from BSD, GNU, and X11. It runs on most (NuBus and OCU bus) Power Macintosh systems; Performa, PowerBook, and multiprocessor ports are currently under development. MkLinux is robust, powerful, freely distributable, and source code compatible with most other Linux systems. It provides a full suite of development tools, support for AppleTalk, HFS, and Objective-C, and access to a vast amount of free software. MkLinux is a great way to "come up to speed" on Mach, UNIX, and Rhapsody.

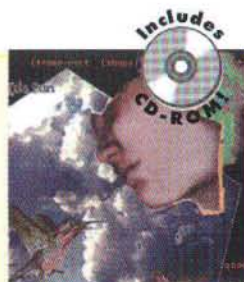
- MkLinux user community supports FTP and web servers, development and porting efforts, and several mailing lists
- The Apple sponsored reference release contains a wealth of introductory and reference material on Linux, Mach, NeXT, and the Power Macintosh
- Includes free 3.0 upgrade  
(BMKLINUX) Our Price **\$50**



**VIP-BASIC:  
Visual Interactive  
Programming in BASIC**  
by Mainstay

Now you can create full-featured, stand-alone Macintosh and Power Macintosh applications in standard BASIC code! VIP-BASIC 2.0 is the fastest way to program your Macintosh.

- Rapid application development environment with application framework, mix and match: VIP-BASIC high-level subprograms
- Import pre-existing BASIC code: automatically integrate BASIC code, export C Code for compiling; automatically convert your BASIC code to C for compilation with Metrowerks' CodeWarrior (SVIPBASIC) Our Price **\$195**



**Macintosh Common Lisp 4.0**  
by Digitool, Inc.

Macintosh Common Lisp provides users with a rich set of object-oriented dynamic language features making it especially well-suited for rapid prototyping, custom development for business and education, scientific and engineering applications, and academic research.

- Power PC native environment & compiler, full Macintosh support

**Check out our Web site!**

• Full product descriptions • Hundreds of more products  
<http://www.devdepot.com>

**Pro Fortran**  
by Absoft Corporation

Absoft Pro Fortran combines native F90, VAX compatible F77, and C/C++ compilers into a single, easy to use environment. All compilers are link compatible and operate through a common interface.

- Graphical debugger, browsers, array display, performance profiler, linker, MRWE application mainframe
- MIG graphics library, Absoft Create Make, several utilities, the latest version of MPW and illustrated documentation
- Whole array operations, modules, interface blocks, and user-defined types or data structures
- Dynamic memory allocation and new control constructs
- F90 is link compatible with Absoft F77, C++, MrC and CodeWarrior
- It is fully compatible with Toolbox, MPW tools, and most third-party products  
(SAPROF) Our Price **\$899**

**Order Toll-free**  
**800-MACDEV-1**  
(800.622.3381)



**VIP-C:  
Visual Interactive  
Programming in C**  
by Mainstay

Now you can create full-featured, stand-alone Macintosh and Power Macintosh applications in just minutes. VIP-C 2.0 is the first rapid application development system for creating complete Macintosh programs in standard ANSI C.

- Includes powerful, tightly integrated visual debugger, Import pre-existing C code: automatically integrate C code with a current project
- Includes full-featured mini database: (ltd to 32K) of the powerful VIP-BASIC database manager gives you everything you need to setup royalty-free, multi-user database applications  
(SVIPC) Our Price **\$295**

- CLOS, the standard Common Lisp object system
- Interactive dynamic environment, multiple processes
- Automatic memory management and self-typing data
- Ephemeral garbage collector, smaller application footprint
- Compiles with Common Lisp industry standard and smart programmable tools, 110+ mb of user contributed code
- Complete on-line documentation (manual sold separately)
- Software license and registration card  
(SMCLISP) Our Price **\$675**



## ObjectMaster Professional Edition

by Altura Software, Inc.



Object Master is an innovative programming environment that provides all the necessary tools to write, organize, and navigate through source code.

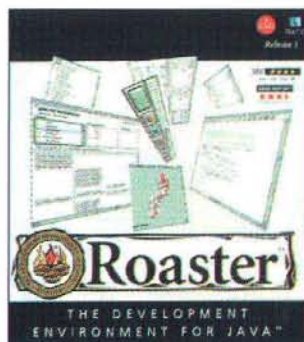
- Write code using the most robust source code editor available on the desktop
- Organize source code into projects to quickly access and manipulate all files
- Navigate through source code using intuitive graphical Browser windows

(SOMPE) Our Price **\$399**

## Roaster

by Roaster Technologies, Inc.

Roaster Release 3 is now available and shipping! Get the most out of Sun's Java™ programming language with this powerful development environment.



- Features include: ability to build completely stand-alone Macintosh applications or applets; visual interface builder; ability to create cross-platform zip files; powerful Java debugger; wizard for quickly creating Java applets or applications; JDBC included; Java object database included; ability to call AppleScripts from Java; Just-in-time (JIT) compiler; JDK 1.0.2 support; class tree and hierarchical class browser; much more!
  - Software includes: Over 300 example applets and applications; Netscape Internet Foundation Classes; Object Design's PSE for Java; OpenLink Software's JDBC Drivers; OpenSpace Java Generic Library; Microline Component Toolkit Lite 3.0; much more!
  - Requirements: Runs on 68k or PowerPC, CD-ROM
  - Price includes all web-based updates
- (SROAST3) Our Price **\$49**

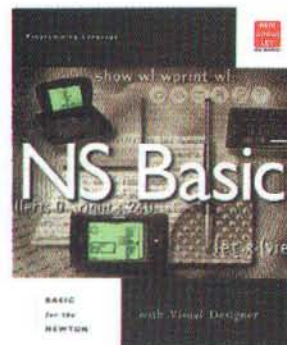
**WAIT...  
There's  
More!**

Here's a list of all available products. For full product descriptions please see our Web site, or feel free to call, fax, or E-mail us.

PRODUCT	CODE	OUR PRICE
LPA MacProlog Developers Edition	SLPAD	995.00
LPA MacProlog Programmers Edition	SLPAP	495.00
LS Fortran Pro	SLSFORT	595.00
LS Fortran Plug-In	SLSFPI	199.00
Mac FORTRAN II	SFORT2	595.00
Power MachTen-UNIX	SM10PPC	695.00
Presenting Magic Cap	BPRESMAGIC	15.25
SmalltalkAgents	SSTA	695.00
Think Pascal 4.0	SPASCAL	165.00

## NS BASIC 3.6 for the Newton with Visual Designer

by NS BASIC Corporation



- A fully interactive implementation of BASIC programming language
  - Runs entirely on the Newton – no host is required
  - Create files, access the built in soups, and the serial port for input and output
  - Work directly on the Newton, or through a connected Mac/PC and keyboard
  - Get the **BASIC Internet Tool**, available at no charge to NS BASIC users from [www.nsbasic.com](http://www.nsbasic.com)
  - Release Notes with sample code are available from the same location
  - Runs on any Newton MessagePad 130 with NS BASIC and the Newton Internet Enabler. Also runs on MP 1201s with NOS 2.0 that have full memory available
  - Write short programs to access News, mail and the web
- (SNSBASIC) Our Price **\$99**



## CodeBuilder

by Tenon Intersystems

CodeBuilder is a powerful and unique Macintosh software development tool for porting existing apps or developing new, advanced applications on Power Macs and Power Mac clones.

- A powerful Macintosh software development tool suite of C, C++, Objective-C, Java, Ada, and Fortran development tools
  - Complete UNIX & X development environment for developing UNIX or Macintosh apps
  - Includes compilers and source-code debugger for Objective C, and C, C++, Ada 95 and Fortran 77
  - Web & internet scripting tools: Perl, MacPerl, tcl/tk, bash, sh, and csh
  - Supports Rhapsody kernel APIs and Rhapsody TCP sockets
- (SMIOCODEB) Our Price **\$149**





## ObjectSet Mail SDK

by Smartcode Software

- Powerful C++ classes for integrating Internet e-mail in your applications
- Helps you write software that can share mail with other leading e-mail products
- Royalty-free MIME, SMTP, and POP3 APIs

- Gives you the most robust MIME parser and encoder available
- Ideal for use in Internet and Intranet environments
- Comes complete with samples with documented, reusable source code
- Free standard technical support (SOSMSDK) Our Price **\$495**

## r-tree Report Generator

by Faircom

Handles virtually every aspect of report generation:

- Complete C source
- Complex multi-line reports
- Multi-file access
- Complete layout control
- Conditional page breaks
- Nested Headers and Footers
- Horizontal Repeats

(SRTGR) Our Price **\$445**



## Spellswell Plus 2.1

by Working Software

- Award-winning, comprehensive, practical spelling checker that works in batch mode or within applications that incorporate the Apple Events Word Services protocol (e.g., Eudora, WordPerfect, Communicate!, and InfoDepot)
- Checks for spelling errors as well as common typos like capitalization errors, spaces before punctuation, double word errors, abbreviation errors, a/an before vowel/consonant, etc.
- MacTech orders include developer kit with Writesswell Jr., a sample AppleEvents Word Services word-processor and its source code
- Available for OEM Sales

(S SPELL) Our Price **\$49**

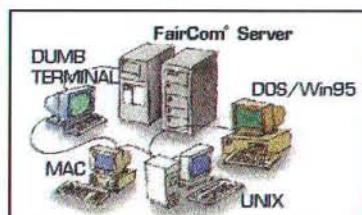
## c-tree Plus® Database Handler

by Faircom

Unsurpassed Cross Platform Tools for Mac Developers!

- Full C Source
- Client/Server Option
- Over 16 years proven reliability
- Concurrent simultaneous access of Mac/PC files
- Superior throughput and performance
- Unparalleled scalability and flexibility
- Fixed/Variable length files

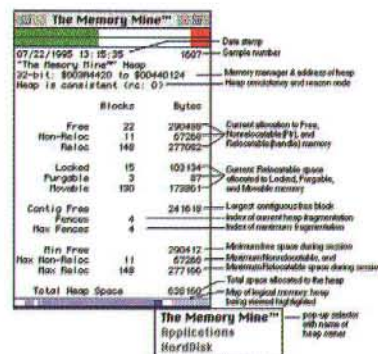
(SCTPDH) Our Price **\$895**



## Memory Mine

by Adianta Inc.

- Monitor heaps, identify problems such as memory leaks, and stress test applications
- Active status of memory in a heap is sampled on the fly: allocation in non-relocatable (Ptr), relocatable (Handle) and free space is shown, as are heap corruption, fragmentation, and more
- Allocate, Purge, Compact, and Zap memory lets users stress test all or part of a program (SMEEMINE) Our Price **\$99**



## ScriptDemon

by Royal Software, Inc.



ScriptDemon is a browser plug-in that, for the first time, allows you to deliver and run AppleScripts from Web pages. The ScriptDemon plug-in will execute the embedded AppleScript code included on a Web page. ScriptDemon painlessly and inexpensively handles many previously impossible tasks, such as:

- Using the Intranet to manage all Macintosh computers on-line
- Using the Internet to install and configure software
- Using the Internet to configure hardware
- Delivering complex sets of files and assembling them on the browsing computer
- Providing interactive education and product support on both the Internet and Intranet
- A perfect companion to LiveCard!

(SDEMON) Our Price **\$949**

## Guide Composer™ 1.2

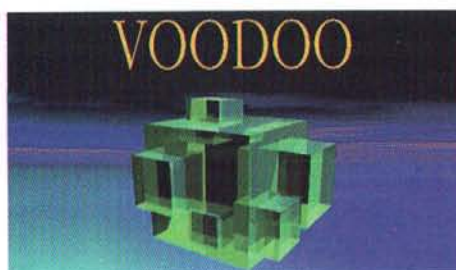
by StepUp Software



- Create powerful Apple Guide help systems for any new or existing Macintosh application
- Provides a WYSIWYG development environment: Guide content is developed in Guide windows
- Design topics, phrases, and panels in the same format as the user will use them
- Features are WYSIWYG interface, Topics, phrases, and hierarchical phrases, Coach marks, Fully-Integrated with Apple's Guide Maker (distributed with Guide Composer), compiles scripts automatically, PICTs in Panels, Generated Guide scripts are modifiable
- FREE Update to all registered Guide Composer users. Demo is available at <http://www.guideworks.com/> (SGCOMP) Our Price **\$99**

SEE RELATED PRODUCTS: AppleGuide Complete, Danny Goodman's AppleGuide Starter Kit, Real World AppleGuide





## VOODOO 1.8

by UNI SOFTWARE PLUS

- Smooth integration with Metrowerks CodeWarrior IDE
- Support of AppleEvents and AppleScript
- Comparison of files of different types (not only text files)
- Configurable local file locking (Finder flag or ckid resource)
- Improve handling of local files including the creation of folder structures
- Significant performance improvements in many places
- Essential parts PowerPC native
- Version control tool for the simple and clear management of projects in which files are created in numerous versions (variants and revisions)
- Allows both variant and revision control, and it manages not only variants and revisions of single files, but of a whole software project (multi files, multi users, multi-variants, access rights, etc.)
- Graphical user interface and is not only suitable for mere source code control but can handle all different kinds of files with amazing compression rates: typical size of delta between arbitrary files 5%
- Please note special prices for multiple copies:  
Single license (SVOODO01) **\$229**; 2 pack (SVOODO02) **\$359**;  
5 pack (SVOODO05) **\$799**; 10 pack (SVOODO010) **\$1369**;  
20 pack (SVOODO020) **\$2399**

Additional pricing available on request.

SEE RELATED CATEGORY: Dev. Environments

## StoneTable 68K/PPC

by StoneTable Publishing

- StoneTable is a replacement for the Macintosh List Manager
- Available for use with Think C, MPW C & Pascal, CodeWarrior C and Pascal
- Includes libraries for 68K and PowerPC
- An LTable-like class is provided to incorporate StoneTable into the PowerPlant environment  
(SSTONEFAT) Our Price **\$199**

## QC

by Onyx Technology

High performance runtime stress testing for applications.

- Tests include heap checks, purges, scrambles, handle/pointer validation, dispose/release checks, write to zero, de-reference zero as well as other tests like free memory invalidation and block bounds checking
- Extremely user friendly – ideal for non-programmer testers
- Also available in Japanese  
(SQC) Our price **\$99**

## AppSketcher 1.0 for BeOS

by BeatWare, Inc.



AppSketcher is the

premier programming tool for the BeOS.

- The fastest way to develop software on the BeOS
- Drag and drop design is great for creating user interfaces
- Supports localization for worldwide sales
- Expands applications easily without manual code modifications
- Shortens development time by automating the Make process through direct communication with the BeIDE (SASFB) Our Price **\$199**



## MacA&D 6.0

by Excel Software

MacA&D combines the capabilities of MacAnalyst plus additional detailed design, code generation and code browsing features into one integrated application. It automates structured analysis and design, object-oriented analysis and design, state modeling, task design, data and screen modeling, code editing and browsing,

reengineering, requirement traceability and a multi-user data dictionary. It generates SQL from data models, C++ or Object Pascal from class diagrams and C, Pascal, Basic or Fortran code from structure charts.

- Structured analysis and design
- Object-oriented analysis and design
- Real-time and multi-task design
- Data and screen modeling
- Integrated code editing and browsing
- Multi-user dictionary and requirements
- Code to design diagrams for C, C++, etc.
- Design diagrams to code for C, C++, etc.
- State modeling diagrams and tables
- Use cases with traceability  
(SMACADP) Our Price **\$1995**

**Order Toll-free**  
**800-MACDEV-1**  
(800-622-3381)

## Apprentice 6 by Celestin Company

**Celestin**

Apprentice 6 is a high-quality CD-ROM collection of over 600 megabytes of up-to-date source code, utilities, and info for Mac programmers. All of the source code and utilities are completely new or updated for this release.

- Frontier 4.1, the highly-acclaimed scripting environment
- More PowerPlant AND many more PowerPC samples
- Cool new languages and environments added (Clean, Eiffel, F, Tcl-Tk)
- Hot new demos from leading Mac development companies  
(SAPPRENT6) Our Price **\$35**



## SoftPolish CD-ROM

by Bare Bones Software

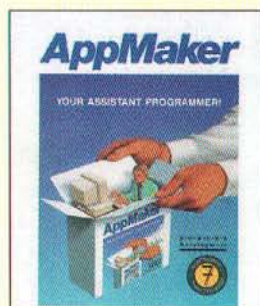
- The essential tool for software quality assurance on the Macintosh
- Helps you identify inconsistencies with Apple's user interface guidelines, misspelled words, missing resources, and other mistakes
- Provides tools to put the finishing touches on software distribution packages prior to release
- Works independently of any programming language or environment
- Ideal for sanity checking software throughout the development process (SSOFTPOL) Our price **\$99**



## AppMaker

by Bowers Development

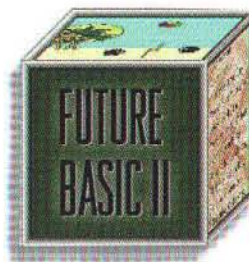
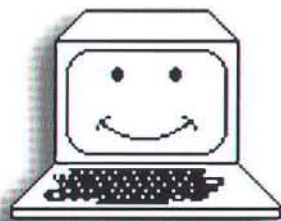
- Develop the user interface for a Macintosh application using the original interface builder
- Just point and click to design your application
- Creates resources and generates excellent source code
- Supports most development environments including Metrowerks, Symantec, or MPW; C, C++, or Pascal; procedural or object-oriented, using PowerPlant, TCL, or MacApp
- The generated code uses the Universal Headers to provide PowerMac compatibility
- Great tool for beginners to learn object-oriented and Macintosh Toolbox programming techniques
- Includes one-year subscription on CD and hardcopy documentation (SAPPMAKE) Our Price **\$299**



## B-Tree HELPER 2.2

by Magreeable Software

- Inexpensive database engine for Macintosh programmers in C source code
- Uses contiguous fixed length blocks
- Expands the file as necessary and contracts files when possible
- Inserts and deletes keys in one or more B-Trees
- Finds keys equal to, less than, or greater than a given value in a few hundredths of a second
- Finds lists of records whose keys are equal to, less than, or greater than a given value or are in a range of values (SBTREE) Our Price **\$149**



## Future Basic II

by Staz Software

FutureBASIC II is the award winning leader in Macintosh BASIC programming.

- Source level debugger and Interactive compiler/editor
- Multi-file Project manager and Multi-file find and replace
- Super fast compilation, 32 bit clean, and System 7.x savvy
- QuickBASIC converter
- Getting Started manual with over 500 example files
- Full support of standard BASIC (SFBASIC2) Our Price **\$229**



dtF

by dtF Americas

- True relational database system for Apple Macintosh computers
- Provides a powerful choice for developers who want to create database centered applications with no performance trade-offs
- Features SQL, full transaction control, error recovery, single user, client server architecture and multi-platform support including DOS, Windows, OS/2 and UNIX
- The C/C++ API is identical and fully portable across all supported platforms
- Third-party vendors supporting dtF will be able to offer a variety of advanced features and benefits to their customers royalty free
- Tools are included for importing, exporting, creating and managing databases and users
- Supported development environments include: Symantec, MPW, Metrowerks and more Mac/SDK (SDTF) Our Price **\$695**

## Step-Up Installer Pack

by StepUp Software

- Package of several Installer "atoms" that let developers incorporate graphics, sounds, file compression and custom folder icons into installation scripts
- Compression formats supported are Compact Pro & Diamond
- Each atom also available separately
- Compression requires additional licensing (SINSTALL) Our Price **\$219**

## ScriptGen Pro

by StepUp Software

- Installer script generator which requires no programming or knowledge of Rez
- Supports StepUp's InstallerPack, Stuffit decompression, Compact Pro decompression, custom packages, splash screens, network installs, and resource installation (SSCRPTGEN) Our Price **\$169**





## Tools Plus libraries + framework by Water's Edge Software

Easily create compact, fast running, professional looking applications and plug-ins\*. Tools Plus lets you create virtually any user interface element with a single routine, and it transparently provides a robust infrastructure to make all your pieces work together as an application. Rated 4 stars by Macworld! MacTech (July 96): "it's an incredibly rich collection of tools... If you are interested in developing applications that have 'quality' written all over them, then Tools Plus is for you."

- Simplifies programming and thins source code
- Automates all standard GUI elements
- Thousands of extras, from floating palettes and tool bars to powerful picture buttons
- Includes numerous 3D grayscale options
- Over 1/2 MB of custom fonts, icons, cursors, and other resources
- Includes SuperCDEFs world-class controls (an \$89 value) free

(STOOLCW) Our Price **\$249**

CodeWarrior Gold (C/C++ & Pascal, 68K & PPC)

(STOOLCWB) Our Price **\$199**

CodeWarrior Bronze (C/C++ & Pascal, 68K)

(STOOLSYM) Our Price **\$199**

Symantec (THINK) C/C++ and THINK Pascal (68K)

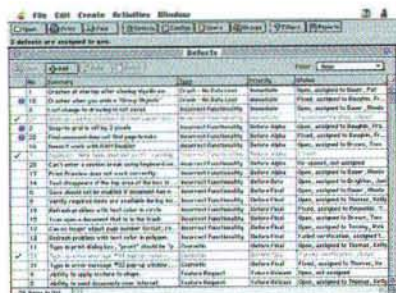
(STOOLSYM) Our Price **\$149**

Symantec (THINK) C/C++ (68K)

(STOOLPAS) Our Price **\$149**

THINK Pascal (68K)

\*CodeWarrior required to write plug-ins



**Order Toll-free  
800-MACDEV-1  
(800-622-3381)**

## TestTrack-Bug Tracking the Macintosh Way by Seapine Software, Inc.

- Tracks bugs, feature requests, test configurations, users, and more
  - Includes notifications, security, a powerful filter mechanism, and multiple reports
  - Links your testers, engineers, documentations staff, and project managers together to ensure all bugs are identified, fixed, and documented
  - Eliminates the need to build custom bug tracking solutions using general purpose database tools
  - Supports single- and multi-user bug databases (additional licenses required to use multi-user features)
- (STETR) Our Price **\$129**

## CodeBuilder by Tenon Intersystems



CodeBuilder is a powerful and unique Macintosh software development tool for porting existing apps or developing new, advanced applications on Power Macs and Power Mac clones.

- A powerful Macintosh software development tool suite of C, C++, Objective-C, Java, Ada, and Fortran development tools.
- Complete UNIX & X development environment for developing UNIX or Macintosh apps
- Includes compilers and source-code debugger for Objective C, and C, C++, Ada 95 and Fortran 77
- Web & internet scripting tools: Perl, MacPerl, tcl/tk, bash, sh, and csh
- Supports Rhapsody kernel APIs and Rhapsody TCP sockets (SMIOCODEB) Our Price **\$149**



## Phyla™: Object-Oriented Database by Mainstay

- Powerful Databases Without Programming: Phyla handles your all your complex database needs
- Define a Database in Minutes: Using an intuitive, graphical user interface
- Objects Are a More Natural Approach:

Phyla creates real world databases

- Drag-and-Drop Ease: Relate objects by simply dragging objects between windows
  - Create Custom Forms and Reports: Quickly create custom forms and reports
  - Fast Finds and Sorts: Perform complex queries and calculations without programming
  - Synchronize Multiple Databases Copies
  - Password Protection With Access Limitations
  - Easy Import and Export: Import from other databases, export data in various formats
- (SPHYLA) Our Price **\$179**



## BBEdit 4.0.4

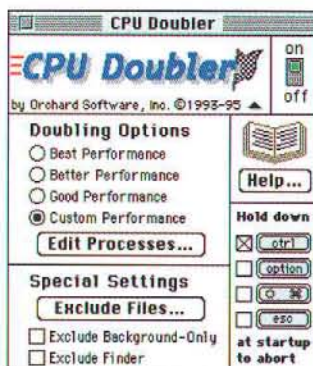
by Bare Bones Software

A powerful, easy-to-learn text editor. Adds new features for HTML coders, including a spelling checker and HTML tag palette. Accelerated for Power

Macintosh; dragging supported everywhere; Internet Config aware; PowerTalk aware. Integrated support for Symantec's IDE, Metrowerks CodeWarrior, THINK Reference 2.x, MPW ToolServer, and most other environments. Many UNIX style tools, including "grep" searches, file comparisons, and sorting multi-file search and replace. PopUpFuncs feature lets you jump to a function from a menu.

(SBBEDIT) Our Price **\$119**





## CPU Doubler by Orchard Software

- Performance enhancement utility for the Macintosh
  - Increases the speed of your computer by 100%
  - Works on both the PowerPC and 68K Macintosh
  - Manages computer throughput using a proprietary scheduling algorithm
  - Ensure optimal performance and compatibility
- (SCPU2X) Our Price **\$79**

SEE RELATED PRODUCTS:  
Development Environments

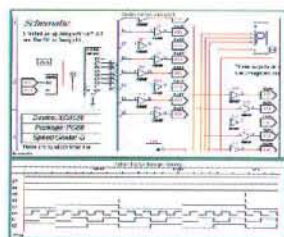
## CompileIt!

by Royal Software, Inc.

CompileIt!, the first HyperTalk compiler, is a complete development system for the creation of XCMDs and XFCNs.

- Expand the capabilities of your environment by using CompileIt! and the ROM Toolbox extensions
- Increase the speed of routines written in HyperTalk by turning scripts into externals
- Protect sensitive code from prying eyes because your code is now compiled!
- Easily learn Macintosh programming by exploring the ROM Toolbox
- Includes DebugIt!, a valuable source-level debugger for externals created with CompileIt!

(SCOMPIT) Our Price **\$149**



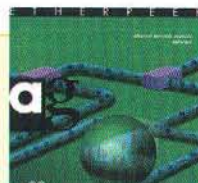
## DesignWorks 4.0 by Capilano Computing

DesignWorks 4.0 has all the ease of use and schematic editing power of previous versions, plus new features designed to make your entire design process easier and more error free. The 4.0

version has new menu customization and scripting features that will directly address your design checking and interfacing needs.

- Flexible schematic editing features speed the drawing process
- Full Undo/Redo on all editing operations
- Hierarchical design with unlimited levels is fully supported
- Powerful attribute features allow arbitrary text information to be associated with any signal, device or device pin
- Extensive symbol libraries with over 12,000 parts in ANSI and IEEE format
- Integrated device symbol editor allows you to create custom symbols using standard drawing tools
- Interactive digital simulator option is available. No netlists, no application switching!

(SDWORKS) Our Price **\$995**



## EtherPeek by AG Group, Inc.



EtherPeek for Macintosh is a full-featured protocol analyzer that allows you to quickly and easily test and debug network communication, and:

- Check for protocol compliance
  - Use hundreds of built-in decodes
  - Develop custom packet decoders
  - Filter packets during or after capture
  - Test device reactions to specific packet types
  - Customize or alter packets for transmission
  - Generate traffic to test varying loads
- (SEPEEK) Our Price **\$745**

## OpenGL for the Macintosh by Conix Graphics



OpenGL is the premier 3D graphics library that allows software developers the ability to develop high-quality, interactive 2D and 3D graphics applications. OpenGL can perform the following wide range of functions which will enhance the development of all graphics software:

- Geometric primitives (points, lines, and polygons)
- RGBA or color index mode
- Viewing and modeling transformations
- Texture Mapping, Lighting, Shading and Z Buffering
- Atmospheric Effects (fog, smoke, and haze)
- Alpha Blending (transparency)
- Antialiasing, Accumulation Buffer, Stencil Planes
- Display list or immediate mode
- Polynomial Evaluators (to support Non-uniform rational B-splines)
- Feedback, Selection, and Picking Raster primitives (bitmaps and pixel rectangles)
- Pixel Operations (storing, transforming, mapping, zooming)

(SOPENGL) Our Price **\$389**



## VText by Vivistar

VText is a C++ add-on library for Metrowerks' PowerPlant application framework. VText provides complete Macintosh text support including: greater than 32kb text, undo, drag and drop editing, AppleEvent scripting and recordability, full support for multibyte characters and inline

input methods including Japanese and Chinese text, and full support for bi-directional script systems including Arabic and Hebrew.

- Full featured text engine for Metrowerks' PowerPlant
  - Stylesets and rulersets with tabs
  - Flexible object oriented C++ API
  - Full undo and drag and drop editing
  - WorldScript savvy including bidirectional and multibyte scripts with inline editing
  - AppleEvent factored for scriptability and recordability
- (SVTEXT) Our Price **\$349**

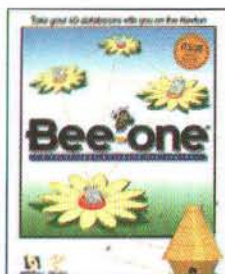




## QUED/M 3.0

by Nisus Software

- The programmer's text editor that defined the industry standard for speed and efficiency
- PowerPC native
- Features integrated support for Symantec C/C++, Metrowerks CodeWarrior 6, and MPW
- Supports all the major development environments on the Macintosh.
- Powerful editing features, including unlimited undo and redo, macro language, scripting, text folding, ten editable/appendable clipboards, markers, displaying text as ASCII codes, dynamic coloring of C/C++ keywords/comments, rectangular and non-contiguous selection
- Includes Celestin Company's APPRENTICE 4 (SQUEDM) Our Price **\$89**



## Bee-one

by Power Box

Bee-one lightens your load on the road by adapting relational databases developed under 4D® to the Newton Platform. Once the program is installed on both the Macintosh and the Newton, it takes 4 simple steps to use Bee-one!

- Database transfer, Set-up, Use, and Synchronization (SBEEONE) Our Price **\$139**

**WAIT...  
There's  
More!**

**Here are more products. For full product descriptions please see our Web site, or feel free to call, fax, or E-mail us.**

### PRODUCT

Fortran 77 SDK  
ICONIX PowerTools-10 Pack  
ICONIX PowerTools-6 Pack  
ICONIX PowerTools-8 Pack  
ICONIX PowerTools-AdaFlow  
ICONIX PowerTools-ASCII Bridge  
ICONIX PowerTools-CoCoPro  
ICONIX PowerTools-DataModeler  
ICONIX PowerTools-FastTask  
ICONIX PowerTools-FreeFlow  
ICONIX PowerTools-Object Modeler  
ICONIX PowerTools-PowerPDL  
ICONIX PowerTools-QuickChart  
ICONIX PowerTools-SmartChart  
ICONIX Training & Consulting  
IMSL Math and Stat Library  
Info-Mac X  
Ionizer Real-Time Spectral Reshaping Tool  
LiveAccess™ 1 User Edition  
LiveAccess™ 1 Developer Edition  
LiveCard  
LJ Profiler  
MacFlow™: Flowchart Design and Development  
Mac Source II  
Nisus Writer 5.0  
Plan & Track™ Product Planning and Management  
Screen Machine  
Spyer  
Visual Café

## AG Author

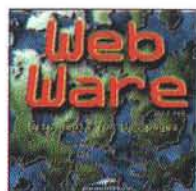
by Lakewood Software

AG Author 1.0 is a full-featured Apple Guide authoring tool with fully customizable project template. The following features are unique to AG Author:

- Support for styled, colored, & hot text
  - Fully customizable project template
  - Flexible compile options
  - Find & replace tool for scripts
  - Multiple open projects
  - Rapid deployment of project globals
- (SAGA) Our Price **\$99**



SEE RELATED PRODUCTS: AppleGuide Complete, Danny Goodman's AppleGuide Starter Kit, Real World AppleGuide



## Web Ware

by BeachWare, Inc.

The ultimate collection of clip media and templates for building your own Web Page. An incredible selection of Shockwave movies, animated GIFs, buttons, bullets, dividers, and sample HTML pages.

There are literally thousands of graphical elements on this disc, all there to spice up your web page. In all, it's about 300 megabytes of creativity only a mouse-click away! System Requirements: PC - 486 or better with 8 MB RAM, Sound card, SuperVGA, CD-ROM drive. Macintosh - Color Mac with 8 MB RAM, CD-ROM drive.

(SWEBW) Our Price **\$24**

### CODE

SF77 589.00  
SICPP10 7,845.00  
SICPP6 5,945.00  
SICPP8 6,945.00  
SICADA 1,395.00  
SICASCII 1,395.00  
SICCOCO 1,395.00  
SICDATAMOD 1,395.00  
SICFASTTASK 1,395.00  
SICFREEFL 1,395.00  
SICOBJMOD 1,395.00  
SICPOWER 1,395.00  
SICQUICKCH 1,395.00  
SICSMART 1,395.00  
TICONIX 2,945.00  
SIMSLSTAT 495.00  
SINFOMAC10 39.95  
SIONIZER 800.00  
SLAUE 69.00  
SLADE 99.00  
SLCARD 149.00  
SLJPROF 295.00  
SMACFLO 179.00  
SMACSOURCE 29.95  
SNISUSW 220.00  
SPLNTRK 179.00  
SSM 24.00  
SSPY 39.00  
SVCAFEMAC 199.00

### OUR PRICE

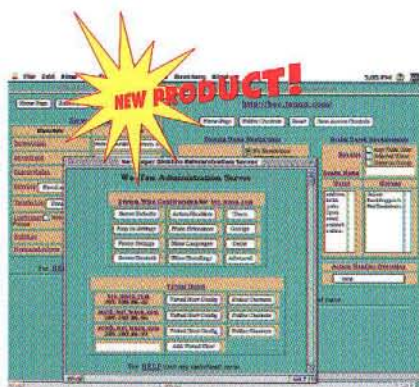


## WebTen

by Tenon  
Intersystems

WebTen is an industrial-strength, high-performance Apache Web server for Power Macs. WebTen's Web-based browser interface enables local or remote administration via your favorite browser. Since Apple's NeXT acquisition, Tenon has extended their unique "UNIX virtual machine" technology to produce a set of "Rhapsody-Ready" internet applications. WebTen is the first offering in this series.

- WebTen is the fastest Web server on Power Macintosh
- Sustains up to 10,000 connections a minute, or over 10 million connections a day
- Apache runs in Tenon's multi-threaded, pre-emptive multitasking environment
- Tenon's unique technology supports the widely acclaimed Apache Web server as a double-clickable Macintosh application (SWEPTEN) Our Price **\$495**



## OOFIE Reporter Writer

by A.D. Software

- Full embedded report-writer, allows you to preview page-by-page and either print or save as plain text, HTML or RTF
- Multiple levels of breaks, database views, headers and footers are provided using a clean object-oriented design
- Includes RAM-based version of OOFIE database. Included in full OOFIE Platform Bundle
- Saving to file without preview of printing is cross-platform-run on your Mac/Win/Unix server and creates web pages
- Price includes 1-year subscription (SOORW) Our Price **\$499**



## WebSiphon

by Purity Software, Inc.

WebSiphon is one of the most anticipated new CGI products for Macintosh Webmasters delivering a complete authoring tool for truly revolutionary sites. This product alone can replace most all CGIs on

your site resulting in increased dynamic serving speed, reliability, and powerful scripting and database abilities directly within your HTML pages! Also includes Verona, the fastest flat-file database server available for Macintosh web sites.

(SWSIPHON) Our Price **\$495**

## PageCharmer: Sizzling Effects...



## PageCharmer 1.0

by Mainstay

PageCharmer is a set of customizable interactive applets that enhance web pages without writing a single line of HTML code. Whether the web site is already up and running or designing one from scratch, PageCharmer gives you the power to make it stand out from the crowd with sophisticated applets that can be personalized to fit most any need.

### FEATURES:

LiveG-Map, LiveT-Map, LiveG-Button, LiveT-Button, LiveGT-Button, LiveG-Ticker, LiveT-Ticker, LiveG-Marquee, and LiveT-Marquee.

(SPGCHRM) Our Price **\$139**



## BBEdit 4.0.4

by Bare Bones Software

A powerful, easy-to-learn text editor. Adds new features for HTML coders, including a spelling checker and HTML tag palette.

Accelerated for Power Macintosh; dragging supported everywhere; Internet Config aware; PowerTalk aware. Integrated support for Symantec's IDE, Metrowerks CodeWarrior, THINK Reference 2.x, MPW ToolServer, and most other environments. Many UNIX style tools, including "grep" searches, file comparisons, and sorting multi-file search and replace. PopUpFuncs feature lets you jump to a function from a menu.

(SBBEDIT) Our Price **\$119**



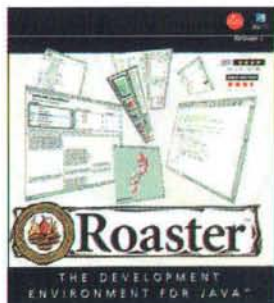
## ObjectSet Mail SDK

by Smartcode Software

- Powerful C++ classes for integrating Internet e-mail in your applications
- Helps you write software that can share mail with other leading e-mail products
- Royalty-free MIME, SMTP, and POP3 APIs for Macintosh, Windows, and Unix

- Gives you the most robust MIME parser and encoder available
- Ideal for use in Internet and Intranet environments
- Comes complete with samples with documented, reusable source code
- Free standard technical support (SOSMSDK) Our Price **\$495**





**Order Toll-free**  
**800-MACDEV-1**  
 (800.622.3381)

## Roaster

by Roaster Technologies, Inc.

Roaster Release 3 is now available and shipping! Get the most out of Sun's Java™ programming language with this powerful development environment.

- Features include: ability to build completely stand-alone Macintosh applications or applets; visual interface builder; ability to create cross-platform zip files; powerful Java debugger; wizard for quickly creating Java applets or applications; JDBC included; Java object database included; ability to call AppleScripts from Java; Just-in-time (JIT) compiler; JDK 1.0.2 support; class tree and hierarchical class browser; much more!
- Software includes: Over 300 example applets and applications; Netscape Internet Foundation Classes; Object Design's PSE for Java; OpenLink Software's JDBC Drivers; OpenSpace Java Generic Library; Microline Component Toolkit Lite 3.0; much more!
- Requirements: Runs on 68k or PowerPC, CD-ROM
- Price includes all web-based updates  
 (SROAST3) Our Price **\$49**

## Power MachTen 4.0.3

by Tenon Intersystems

MachTen is the only Macintosh product that can turn your Macintosh into a complete Unix workstation. Based on BSD4.4 and the Mach kernel, MachTen brings the power of Unix to your desktop at an extremely attractive price point. MachTen enables you to:

- Run a high speed internet server, complete with WWW, FTP, NFS, DNS and print service
- Build a Multihomed Web Server
- Develop applications in a Unix development environment, replete with the acclaimed GNU development toolset
- Program in Ada, C, C++, Pascal, Fortran, and more
- Run Xwindows applications, from remote workstations or on your Macintosh
- Run hundreds of Unix applications, already ported for MachTen and available on our Ported Applications CD-ROM
- Run Software.com Inc's acclaimed Post.Office mail transport service (SM10PPC) Our Price **\$695**

## ScriptDemon

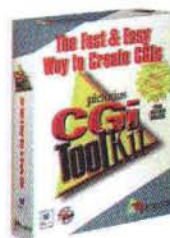
by Royal Software, Inc.



ScriptDemon is a browser plug-in that, for the first time, allows you to deliver and run AppleScripts from Web pages. The ScriptDemon plug-in will execute the embedded AppleScript code included on a Web page. ScriptDemon painlessly and inexpensively handles many previously impossible tasks, such as:

- Using the Intranet to manage all Macintosh computers on-line
- Using the Internet to install and configure software
- Using the Internet to configure hardware
- Delivering complex sets of files and assembling them on the browsing computer
- Providing interactive education and product support on both the Internet and Intranet
- A perfect companion to LiveCard!

(SSDEMON) Our Price **\$949**

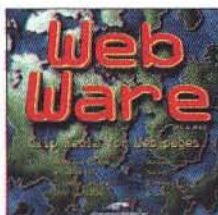


## CGI Toolkit

by Pictorius Inc.

The Pictorius CGI Toolkit is the fast and easy route to high performance CGIs and ACGIs for your Mac Web site.

- Interactively develop CGIs while the web server, the CGI Toolkit and the browser are running on the same machine
- Interactively develop, test and debug CGIs before compiling
- Powerful debugger allows you to edit code, roll back, code and change input values while your application is running
- Fully object oriented so you can re-use your code
- Automatic handling of Apple Events so you can concentrate on building functionality
- Easy creation of multi-function CGIs which reduces application footprint and RAM usage  
 (SCGITLKT) Our Price **\$149**



## Web Ware

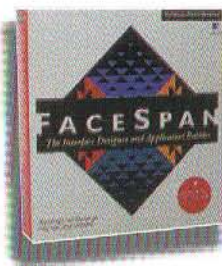
by BeachWare, Inc.

The ultimate collection of clip media and templates for building your own Web Page. An incredible selection of Shockwave movies, animated GIFs, buttons, bullets, dividers, and sample HTML pages. There

are literally thousands of graphical elements on this disc, all there to spice up your web page. In all, it's about 300 megabytes of creativity only a mouse-click away! System Requirements: PC - 486 or better with 8 MB RAM, Sound card, SuperVGA, CD-ROM drive. Macintosh - Color Mac with 8 MB RAM, CD-ROM drive.

(SWEBW) Our Price **\$24**





## FaceSpan v2.1

by Digital Technology International

- Develop integrated software, make stand alone applications, create friendly interfaces
- Develop quick prototypes, print multiple pages with sophisticated layouts
- Script essential elements of the FaceSpan application - Enhanced save options

- Play and record sounds as either "snd" resources or as "AIFF" files
  - Create miniature or complete apps that run on either Power PC or 68k computers
  - Use precise time measurement for implementing timed behaviors — New properties
  - Proportionally scale PICT images -Align images in a pictbox - Automate any application
  - Monitor and respond to low-memory situations-Increased support for Frontier UserTalk!
- (SFACESPAN) Our Price **\$299**

## PreFab Player

by PreFab Software, Inc.

PreFab Player is a faceless background application (similar to a system extension) that lets your scripts query and control otherwise non-scriptable applications, desk accessories and control panels. Player adds verbs that directly manipulate the Macintosh user interface: choose from menus & pop-ups, select radio buttons, type text, determine the name of the frontmost window, the state of a check box, etc. Balloon help identifies non-standard dialog items.

- Adds verbs to AppleScript and to Frontiers UserTalk
  - Controls the Frontmost Application
  - Balloons Identify User Interface Objects
- (SPPLAYER) Our Price **\$95**



## Scripter 2.0

by Main Event Software

For professionals, for novices, for webmasters, for solutions providers, there's only one serious choice. Scripter!

- Scripter and FaceSpan work together: one click opens your FaceSpan script in Scripter, another sends it back

- Debug handlers without modifying your scripts using the Call Box
  - Applet simulation, live editing, Object map, associated terminology
  - Search backwards, block generators, more navigation shortcuts, more drag-and-drop, and an even more enhanced trace log
  - Now Includes ScriptBase; stores your data and media elements and share them between scripts all with a special new browser
  - Easily write and compile scripts that have handler declarations and other vocabulary specific to a particular scriptable application
  - Scripter is the natural companion to AppleScript for users at all levels of proficiency. Don't write scripts without it!
- (SSCRIPTER) Our Price **\$199**

## AppleScript Software Development Toolkit 1.1

by Apple Computer, Inc.

- AppleScript language, system software extension, and script editor
- FaceSpan 1.0
- Developer's redistribution license for AppleScript System software extension and FaceSpan runtime code (SASDT) Our Price **\$49**



## Script Debugger

by Late Night Software Ltd.

- A powerful and flexible AppleScript authoring tool — get the most from AppleScript!
  - Advanced debugging environment offers single-step script execution with breakpoints
  - Script Debugger dictionary browser features a graphical view of objects provided by scriptable applications
  - Includes Late Night Software Scripting Additions — a collection of more than 70 new AppleScript commands, and Scheduler, a utility that allows you to launch scripts at pre-determined times
- (SDEBUG) Our Price **\$129**



## TCP/IP Scripting Addition

by Mango Tree Software

- Award-winning AppleScript scripting addition
- Allows you to write scripts using MacTCP™ commands in AppleScript™
- Send e-mail or files through a script, check if users are logged on (via Finger), automate FTP, Gopher, NetNews, Telnet, and LPR, verify links in HTML documents, and quickly write many other TCP/IP client-server programs
- Works with AppleScript, MacTCP 2.0.4 and Open Transport (STCP) Our Price **\$49**

## WindowScript

by Royal Software, Inc.

WindowScript is the ultimate tool for designing Macintosh user interfaces using HyperCard. Design Real "Macintosh" user-interfaces right inside HyperCard. Until now you either created HyperCard stacks or Macintosh applications. With WindowScript you can literally bring the look and feel of a real Macintosh user-interface to HyperCard. If you're a HyperCard developer, interface designer, application developer, program manager or tester searching for a prototyping tool, WindowScript is perfect for the job.

(SWSCRIPT) Our Price **\$149**





## Apple Media Tool Programming Environment 2.1

by Apple Computer, Inc.

- This object-oriented language and application framework allows programmers to customize features used within the Apple Media Tool authoring environment
- Includes an expanded Apple Media Language (AML) class library, incremental compiling and linking of AML code, faster debugging facilities, Macintosh Programmers' Workshop (MPW), and user-oriented documentation written from an AMTPE developer's perspective
- Portable across 68K, Power Macintosh, and Windows platforms

(SAMTPE) Our Price **\$995**



## Virtual Reality Programming with QuickTime VR 2.0

by Apple Computer, Inc.

- Virtual Reality Programming Book/CD-ROM for QuickTime VR 2.0
- Enables you to write C and C++ programs using QuickTime VR 2.0
- Allows QuickTime VR to be used in games, multimedia titles and other programs
- QuickTime VR 2.0 objects can be zoomed in on, panned, or linked with hot spots
- Both panoramas and objects have hot spots linked to World Wide Web URLs (SVRPQT) Our Price **\$49**

## QuickTime Developer's Kit 2.0

by Apple Computer, Inc.

- QuickTime 2.0 Extension, QuickTime Power Macintosh Extension, and QuickTime Musical Instruments extension
- Utilities like MoviePlayer 2.0, 16-bit Audio Compression, etc.
- Sample content such as MPEG Movies, Music Movies, Time-Code Movies, and 60 field per second movies
- Includes software-only playback features such as faster 2x playback mode for current compressors, Apple Cinepak compressor, 1-bit fast dithering, network tuning, load-into-RAM option, and Photo CD support

(SQTDK) Our Price **\$99**

## QuickTime VR 2.0 Authoring Tools Suite

by Apple Computer, Inc.

- QuickTime VR is a cross-platform software from Apple which enables webpage designers and professional developers to create new multimedia products and webpages incorporating QuickTime VR content. With QuickTime VR, users interactively navigate through 360° views of space, and explore three dimensional objects on Macintosh or Windows-based personal computers
- The QuickTime VR Authoring Tools Suite is a set of Macintosh tools to create and link panoramas and objects from

photographic, digital, video, or computer generated images

- Included is a complete set of documentation for planning, designing, photographing, and creating QuickTime VR panoramas and objects. The authoring tools also allow you to link objects to panoramas using clickable hot spots

Included on the CDs are:

- A software tool (MPW-based) that stitches and blends adjacent images into a panoramic PICT file
- A software tool (MPW-based) that dices and compresses panoramic PICT files to less than 100 KB (low resolution) per panorama

- A scene editor (HyperCard-based) to create QuickTime VR scenes by adding and positioning nodes, hot spots, linking nodes together, and for linking QuickTime VR objects to scenes
- A variety of utility tools for formatting the data into the runtime software

Due to a revolutionary distortion-correcting algorithm, QuickTime VR panoramas and objects maintain a normal perspective when the user moves the mouse. The speed of the algorithm allows up to 24-bit color images. Both vertical and horizontal panning can occur at fast speeds.

(SQTVRATS) Our Price **\$395**

## Multimedia Authoring with Apple Media Tool

by Apple Computer, Inc.

Apple Media Tool offers new multimedia users a way to get started creating interactive multimedia with minimal learning time. This self-paced tutorial will make Apple Media Tool (AMT) even easier to understand and to use. Using this tutorial, you will create a realistic multimedia project using exciting techniques such as QuickTime movies, animation and more. A demo version of AMT is included and can be used for the exercises. Training Format: Tutorial with labs.

(SMWAMT) Our Price **\$49.95**



## Clip VR™

by eVox Productions

Clip VR™ is a new digital image library offering high quality Photographic Virtual Reality (PVR) images for use with QuickTime™ VR and other desktop VR tools.

Clip VR™ Panoramic Image components include alpha channel masks. Combine elements into a composite panorama which is converted to the finished QuickTime VR movie using Make QTVR Panorama Tool. The Components ("Clips") include complete 360 degree scenes, libraries of 360 degree terrains, 360 degree skies, buildings, and objects. Images are provided as .PICT files AND QuickTime VR movie files. Clip VR™ allows you to create high quality VR worlds from pre-photographed component images. Clip VR™ can be used to add excitement to a web site, to increase the interactive value of a CD-ROM, or simply for fun! Requires imaging editing program such as Adobe Photoshop™ to perform .PICT image compositing.

(SCLIPVR) Our Price **\$89.95**





## Media Cleaner Pro

by Terran Interactive

Use Media Cleaner Pro 2.0 to optimize and compress video for CD-ROM, kiosk, or the Internet. Media Cleaner Pro automates your work flow allowing you to get the highest quality video, faster and easier than any other program on the market.

- Includes Adobe Premiere Export module
- Optimal palette generation, Drag-and-drop batch processing
- RealMedia, VDOlive and improved QuickTime support
- Dynamic Preview Window, the Media Wizard, multiprocessor support and more!

System Requirements:

68040 Mac or better (PowerPC strongly recommended, req'd for RealMedia), QuickTime 2.0 or later (2.5 strongly recommended) 8 Mb application RAM, MacOS 7.0.1 (7.5 or later recommended) SoundManager 3.2, CD-ROM Drive

(SMCP) Our Price **\$359**

Registered owners of Movie Cleaner Pro 1.3 or earlier can upgrade (SMCPUP) Our Price **\$129**



## Music Tracks

by BeachWare, Inc.

A new PC/Mac & Audio multimedia music CD-ROM. The clips include musical introductions, fanfares, background music, and more. This collection offers you 100 music clips stored in .WAV format for Windows, SoundEdit & AIFF formats for Macintosh and as Audio tracks for audio CS players. All of the music clips are completely

license and royalty-free!! Mac System requirements: Mac Plus or greater, CD-ROM drive. PC system requirements: Windows 3.1 or later, Sound Blaster compatible board, CD-ROM drive.

(SMT) Our Price **\$24**



## Captivate 4.6:

### Essential Graphics Utilities

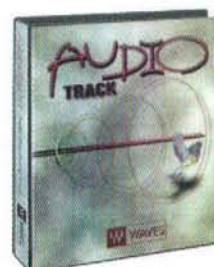
by Mainstay

Captivate™ 4.6 is a powerful collection of graphics utilities for Macintosh, based on Mainstay's acclaimed screen capture utility, graphics and multimedia scrapbook, and

graphics viewer. Captivate provides essential graphics utilities to the professional and hobbyist alike.

- Package includes: Captivate Select, Captivate View, and Captivate Store
- Any one of the three can be used alone, and together they make an unbeatable team
- Whether writing a training manual, creating an ad, or just creating a startup screen from your favorite picture, Captivate is everything professionals need at a price anyone can afford.

(SCAPTIV) Our Price **\$79**



## AudioTrack

by WAVES

AudioTrack is a software plug-in for native processing on digital audio recording and editing systems. Waves

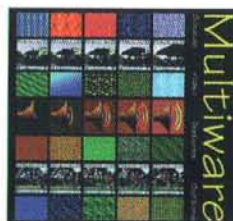
AudioTrack combines the most-needed audio processors into a single piece of software, including 4 bands of equalization, compression/expansion, and noise gating. AudioTrack is ideal for preparing audio for InterNet streaming formats, processing individual mono/stereo tracks of audio and audio for video editing systems including digital sequencers.

Feature list

- A single window interface
- 4-band ParaGraphic Equalizer, Compressor/Expander and Gate
- Instantaneous A/B comparisons of on-line settings
- Pretested setup libraries supplied for various processing solutions
- Power Macintosh native processing (requiring no DSP board)
- Volume and gain reduction meters
- Peak hold and clip meters

Requirements:

The AudioTrack is compatible with all machines supported by Deck II, SoundEdit 16, Adobe Premiere 4.0 and Cubase VST 3.1 (SAUDIOTRK) Our Price **\$270**



## MultiWare

### Multimedia Collection

by BeachWare, Inc.

Introducing a new Audio multimedia music CD-ROM for the Macintosh. This disc is a collection of clips ideal for Desktop Presentations and other Multimedia applications. This incredible collection of license-free media clips is bursting with 240+ color pictures and backdrops (PICT), 200+ sound & music clips (SoundEdit), 140+ QuickTime movies, and a variety of multimedia tools for use with the Macintosh.

(SMWMC) Our Price **\$24**

**Order Toll-free**  
**800-MACDEV-1**  
(800-622-3381)





## Casino!

by BeachWare, Inc.

Can't make it to Vegas this month? Your best bet is Casino! Whether your favorite is Slots, Poker, Blackjack, or Keno, this virtual casino will entertain you for hours with its ten different machines. Mac System

requirements: Color Mac, CD-ROM drive, 4 MB of RAM. PC system requirements: Windows 3.1 or later, CD-ROM drive, 4 MB of RAM.

(SCAS) Our Price **\$24**



## Abuse

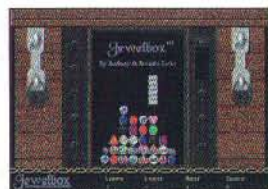
by Bungie Software

Abuse is 360 degrees of side-scrolling action. Run, jump, fall and fly in any direction - through industrial corridors, caverns and sewers. Destroy enemies in any direction with grenade launchers, rocket launchers, napalm and nova spheres! Avoid deadly traps with jet packs and turbo boost!

Key Features:

- Make your own mayhem with the Level Editor
- Hunt down your friends in 8-person multiplayer games
- Awesome Arsenal. Napalm Bombs, Nova Spheres and the Death Saber: just a few ways to lay waste!
- Point and Kill Interface. Move and annihilate mutants in complete 360° freedom
- Blast your way through floors, walls and ceilings in search of the ultimate power-up!
- Abuse is 360 degrees of side-scrolling action. Run, jump, fall and fly in any direction - through industrial corridors, caverns and sewers
- Destroy enemies in any direction with grenade launchers, rocket launchers, napalm and nova spheres! Avoid deadly traps with jet packs and turbo boost!

(SABUSE) Our Price **\$51**



## 1000 Games for Macintosh

by BeachWare, Inc.

The best Macintosh game disc in the entire world, this CD-ROM contains over

one thousand great shareware and public domain programs. Battle ugly aliens, blast apart run-away asteroids, deal yourself that royal flush or solve that 3-D puzzle, this disc has it all! System requirements: Mac Plus or greater, CD-ROM drive, and 2 MB of available RAM (4 MB of RAM when running under System 7).

(STGM) Our Price **\$17**



## Classic Arcade

by BeachWare, Inc.

Ten of your favorite coin-arcade games, redone with killer graphics and sounds! Walk through a virtual arcade and test your game playing skills with these exciting arcade classics. Ten games, including Moon Lander,

Astro-Boing, Hyper Hockey, Ballistic Avenger, and more. System

Requirements: Mac - Color Mac with 8 MB RAM, CD-ROM drive. PC 486 with 8 MB RAM, Sound card, SuperVGA, CD-ROM drive.

(SCLA) Our Price **\$29**



## Marathon Trilogy Box Set

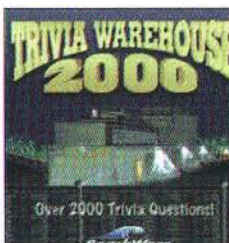
by Bungie Software

The Marathon Trilogy Box Set brings all three Marathon games together in one affordable package, with tons of extras thrown in. Besides Marathon, Marathon 2: Durandal and Marathon Infinity, you'll also receive a staggering 1200 maps, featuring never-released Bungie maps and the winners of the Infinity Mapmaking Contest, The Marathon Scrapbook

(a behind-the-scenes look at themaking of the Marathon games), Marathon collectables like the Marathon 3-sticker set, and to top it off, the award-winning game that laid the groundwork for Marathon: Bungie's breakthrough Pathways Into Darkness.

The Marathon Trilogy Box Set is native to the Power Macintosh, utilizes the graphics acceleration of 630 and 6200 machines, is 8, 16 and 24-bit color capable, and can be played with joysticks and game pads. The package requires a 68040 or higher Macintosh, CD-ROM drive, 8-bit color monitor (13" recommended), and System 7 or later.

(SMTBS) Our Price **\$65**



## Trivia Warehouse 2000

by BeachWare, Inc.

Introducing a fun new PC and Mac CD-ROM. This disc contains two thousand trivia questions, in 45 categories, in several game formats! Test your memory with the Q&A, Concentration, and Multiple Choice games! Categories include:

Animals, Bodies, Bond, Bugs, Cities, Comics, Geography, Gilligan, Gross, Health, Holidays, Horrors, Kids, Knot's Landing, Math, Movies, and many more. Mac System requirements: Color Mac, CD-ROM drive, 4 MB of RAM. PC system requirements: Windows 3.1 or later, CD-ROM drive, 4 MB of RAM.

(STW2K) Our Price **\$24**

**WAIT...  
There's  
More!**

Here are more products. For full product descriptions please see our Web site, or feel free to call, fax, or E-mail us.

### PRODUCT

3D Game Machine  
A Zillion Sounds  
Night Sky Interactive  
Vitamin

### CODE

S3DGAME  
SAZS  
SNSI  
SVITAMIN

### OUR PRICE

**\$299.00**  
**24.00**  
**24.00**  
**24.00**





## System 7.5 Technologies by Apple Computer, Inc.

- Self-paced course designed to allow software developers to write code that extends the functionality of an application for System 7.5
- Contains comprehensive materials for drag-and-drop, threads, standard mail package, and QuickDraw GX printing

Students should be familiar with the basics of developing an application on the Macintosh. Metrowerks CodeWarrior Lite is included on the CD with the lab exercises. The lab assignments were developed in CodeWarrior 8. The labs can also be done in another development environment but project files for them are not provided.

### Training Format: Tutorial with labs.

Requirements: Macintosh or Mac-OS compatible computer with a 68020 processor or greater (PowerPC preferred); 8 MB RAM; 25 MB hard disk space; System 7.5 or later; CD-ROM drive.

(SSYS TECH) Our Price **\$49.95**



## Apple Guide Integration by Apple Computer, Inc.

- Self-paced overview teaches you when and how to add Apple Guide help to your program.
- Powerful help system that can guide the user through a task.
- Tutorial will lead you through the steps necessary to integrate Apple Guide into your application. CodeWarrior Lite is included on the CD with the lab exercises.

### Training Format: Overview with labs.

Requirements: Macintosh or Mac-OS compatible computer with 68020 processor or greater, PowerPC preferred; 8 MB RAM; 25 MB hard disk; System 7.5 or later; CD-ROM drive.

(SAGI) Our Price **\$49.95**



## Virtual Tutor for QuickTime VR

by Apple Computer, Inc.

- Self-paced, hands-on course, which provides a comprehensive environment for learning the steps of the QTVR development process. The student can cover all of the topics or choose areas to focus on. Topics covered include: QTVR capabilities and key concepts, panoramic movies, object movies, QTVR Scene movies and authoring with QTVR
- CD-ROM contains lots of useful examples and demos. In addition to all the step-by-step exercise files

If the student completes the entire course, he/she will create a complete, authored multimedia project similar to the demonstration title that comes on the enclosed CD-ROM. There are approximately 3-4 days of training.

### Training Format: Tutorial with labs.

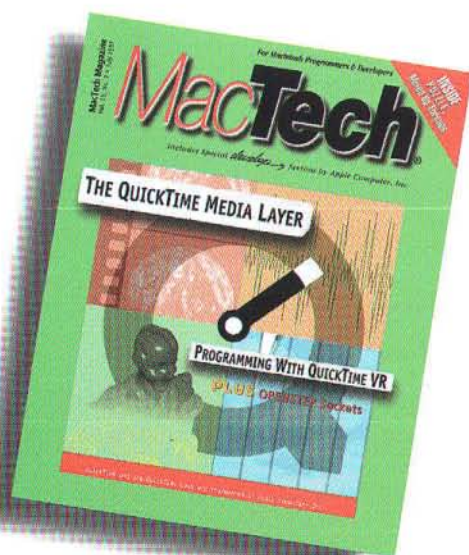
Requirements: 40 MB RAM minimum, 64 MB preferred; Macintosh or Mac OS-compatible computer with a 33 MHz 68040 processor or greater; System 7.1 or later; CD-ROM drive; 17" color monitor.

(SVTFQTVR) Our Price **\$79.95**

*develop* →

Apple's develop™  
back issues are  
available for only  
\$10 per copy.  
To place your  
orders, call

**800/MACDEV-1.**



For Macintosh  
Programmers & Developers  
**MacTech**  
M A G A Z I N E

## MacTech® Magazine

MacTech keeps Mac programmers & developers up to date with everything they need to know about software development. Topics like Rhapsody, Java, QuickTime, OPENSTEP, Objective-C, C/C++, Object Oriented Technologies, product reviews and much more!

Subscriptions:

(MTYRDM) US/Domestic **\$47** for 12 issues

(MTYRCM) Canadian **\$59** for 12 issues

(MTYRFM) International **\$97** for 12 issues

**Back Issues: \$10** each plus shipping (subject to availability)





**CodeWarrior Wear**  
You live it, you breath it... you  
might as well wear it! (XL only)



**Alien T-Shirt**  
(AALIEN) Our Price **\$9.95**

**Arnold T-Shirt**  
(ACWARNLD) Our Price **\$9.95**

**Arnold Jr. T-Shirt**  
(AARNOLDJR) Our Price **\$9.95**

**Blood, Sweat & Code T-Shirt (SS)**  
(ACWSBLOOD) Our Price **\$9.95**

**Blood, Sweat & Code T-Shirt (LS)**  
(ACWLBLOOD) Our Price **\$14.95**

**CodeWarrior Baseball Cap - Black**  
(ACWBHAT) Our Price **\$14.95**

**CodeWarrior Baseball Cap - White**  
(ACWWHAT) Our Price **\$14.95**

**CodeWarrior Sweatshirt - Black**  
(ACWSWEAT) Our Price **\$29.95**

**CodeWarrior Hawaii Five-0 Shirt**  
(ACWHAWAII) Our Price **\$9.95**

**CodeWarrior Winter Hat**  
(AWINHAT) Our Price **\$14.95**

**Debugger Boxer Shorts**  
(ADBOXER) Our Price **\$16.95**

**Discover Programming T-Shirt**  
(ADISCP) Our Price **\$9.95**



**Legtop PODEUM**  
by Rach Inc.

A combination working  
platform and carrying case  
that allows laptop owners a  
safe and comfortable way  
to use their computer in a variety  
of mobile and field environments.

- Straps to your leg for mobility!

(ALGTPPOD) Our Price **\$79**



**MacTech® Mouse Pad**

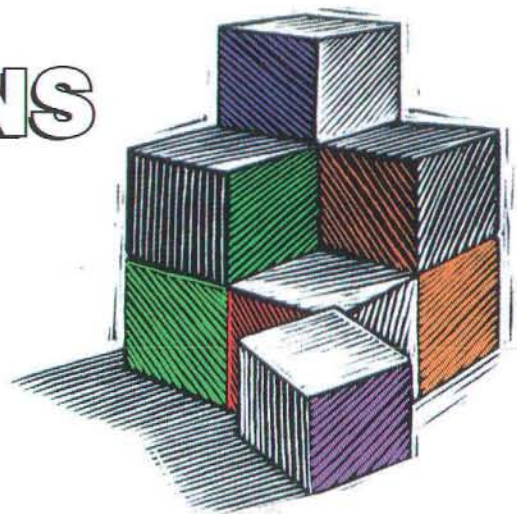
Slide on this! With an extra-large surface (11" by 10") and  
a deluxe sleek plastic coating, you'll be zooming across  
your screen in no time at all. Speed limit not enforced!

(AMTPAD) Our Price **\$8.95**





# PUBLICATIONS



## Getting Started With WebObjects

by NeXT Software, Inc.

If you're a first time user, start here to learn the basics of how to create and run WebObjects applications.

(BGSWO) Our Price **\$14**

## WebObjects Developer's Guide

by NeXT Software, Inc.

A guide to building and understanding WebObjects applications. Takes a close look at the WebObjects scripting language. Additional sections explain the WebObjects architecture and tells you how to integrate your code into the request-response loop, create reusable components, create client-side components, take advantage of powerful Foundation Framework features, and more. Filled with example code. For all WebObjects programmers.

(BWODG) Our Price **\$16**

## D'OLE Developer's Guide

by NeXT Software, Inc.

Distributed OLE is available today, and this book shows you how to use it. Using real-world examples, the book steps you through the process of making your OpenStep objects available on Windows through OLE.

(BDOLEDG) Our Price **\$22**

## Discovering OPENSTEP, Mac

by NeXT Software, Inc.

Introduces programmers to NeXT's OPENSTEP 4.0 Developer product by guiding them through the creation of three applications of increasing complexity. The tutorials demonstrate and explain programming techniques, Objective-C fundamentals, common APIs, and usage of the development tools. Along the way they present summaries of important concepts and paradigms. The book also includes a chapter directing readers to programming resources, further information, and services such as training and support. An appendix offers a concise discussion of object-oriented programming.

(BDOSTEPM) Our Price **\$15**

## Discovering OPENSTEP, Windows

by NeXT Software, Inc.

Discovering OPENSTEP provides an introduction to OPENSTEP programming on Windows NT. It guides the reader through the creation of three applications of increasing complexity. Along the way, it explains concepts and illustrates aspects of Objective-C, OpenStep classes, the development environment, and programming techniques. A short appendix offers a summary of object-oriented programming.

(BDOSTEPW) Our Price **\$16**

## Object-Oriented

### Programming and Objective C

by NeXT Software, Inc.

An introduction to the principles of object-oriented programming in OPENSTEP and the official description of the Objective-C language. Objective-C is easy to learn and use because it adds very little syntax to the C programming language. It's dynamic nature allows you to accomplish things not possible in most other object-oriented languages. For any OPENSTEP programmer.

(BOOPOC) Our Price **\$24**

## Working w/ Interface Builder (for EOF)

by NeXT Software, Inc.

A hands-on, award-winning book designed to help you get your job done with the updated Interface Builder, released with NEXTSTEP 3.3 and the Enterprise Objects Framework 1.1. For any programmer using Interface Builder to design objects that truly work in NEXTSTEP.

(BWIB) Our Price **\$24**

## Using EOF 2.1 w/ OPENSTEP (Mac & Windows)

by NeXT Software, Inc.

Using Enterprise Objects Framework with OPENSTEP describes how to create an Enterprise Objects Framework application on OPENSTEP. It includes a tutorial and a chapter on creating a user interface for an OPENSTEP Enterprise Objects Framework application.

(BUEOFO) Our Price **\$14**

## EOF Developer's Guide for EOF 2.1 (Mac & Windows)

by NeXT Software, Inc.

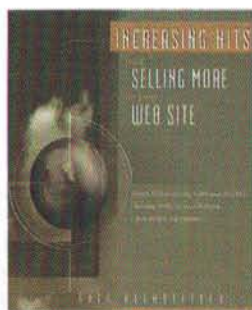
The Enterprise Objects Framework Developer's Guide describes how to develop database applications using the Enterprise Objects Framework tools and classes. It includes an architectural overview of the product, and descriptions of programming tips and techniques. An appendix offers a summary of Entity-Relationship Modeling.

(BEOFDG) Our Price **\$24**

EOF Developer's Guide for EOF 2.0 (BEOFDG20) Our Price **\$24**

EOF Developer's Guide for EOF 1.x (BEOFDG1X) Our Price **\$24**



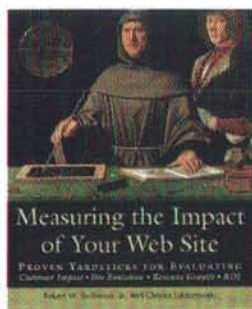


## Increasing Hits and Selling More on your Web Site

by Greg Helmstetter

Written especially for entrepreneurs, corporate marketing managers, small business owners, and consultants, this valuable guide gives you rare tips and tricks you need to know to make your site a commercial success.

(BIHSMWS) Our Price **\$22.45**



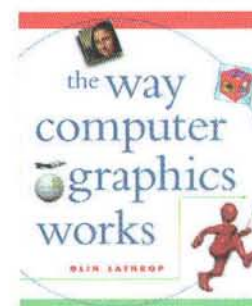
## Measuring the Impact of Your Web Site

by Robert W. Buchanan and Charles Lukaszewski

This book features case studies from many successful and some less successful corporate web site pioneers. You will learn techniques and commercially available tools for

measuring site traffic and visitor behavior—and help you choose the right ones for the job. Written by leading corporate site consultants this book tells you how to develop a management strategy geared toward optimizing web site productivity.

(BMIYWS) Our Price **\$26.95**



## The Way Computer Graphics Works

by Olin Lathrop

A complete guide to mastering computer graphic basics. It is written in a frank, down-to-earth style covering everything from how computer graphics are different from fine art and photographs, to modeling, pixels, and the principles of animation.

All of this is done without resorting to mind-numbing equations and impenetrable technical jargon.

(BWCGW) Our Price **\$29.65**



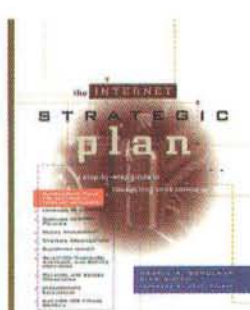
## Debugging Macintosh Software with MacsBug

by Konstantin Othmer and Jim Straus

MacsBug, from Apple Computer, Inc., is the leading debugging software program for the Macintosh. This book/disk package is an all-in-one kit

for using MacsBug. Chapter 1 introduces MacsBug and describes the contents of the rest of the book. Chapter 2 describes how to install MacsBug and enough low level details about the Macintosh so that you can use MacsBug. Includes MacsBug 6.2 on disk.

(BDMSWM) Our Price **\$31.45**



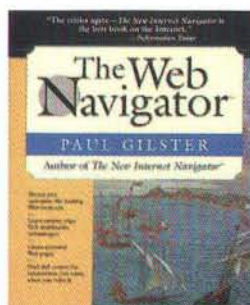
## The Internet Strategic Plan

by Martin A. Schulman and Rick Smith

This book gives you all of the management tools you need for creating a strong Internet and Web presence. A blueprint for your strategic plan, this book guides you every step of the way in establishing your company's

place on the Internet and World Wide Web.

(BTISP) Our Price **\$22.45**

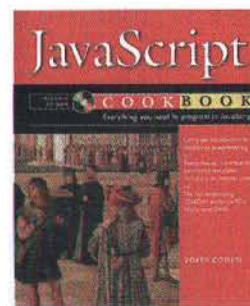


## The Web Navigator

by Paul Gilster

This book keeps you right on top of all the recent changes in the Web, tells you what's out there right now and what's coming in the future. You'll get samples of various Internet sites and candid discussions of providers. You'll receive proven strategies for finding and managing information.

(BTWN) Our Price **\$22.45**



## JavaScript Cookbook

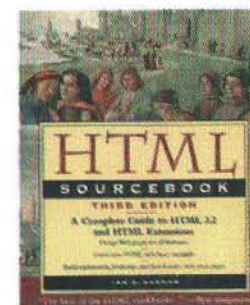
by Yosef Cohen



Everything you need to master the fundamentals of JavaScript programming is in this book/CD-ROM package. A special easy-to-use reference section offers detailed analysis and examples of objects, properties, event handlers, and

statements. You'll also find all the latest information relevant to JavaScript programming since the advent of Navigator 2.0.

(BJSCB) Our Price **\$44.99**



## HTML Sourcebook, 3rd Edition

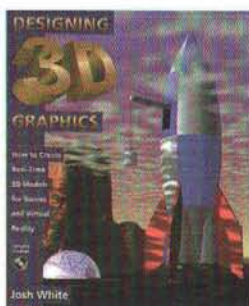
by Ian S. Graham

Critics everywhere agree, HTML Sourcebook is the best guide to HTML for Web professionals. That's because no other book makes it so easy for you to quickly master all the commands, tools, and expert techniques you need to create cutting-edge Web page

documents. Completely revised and expanded by nearly 50 percent, this new third edition of the best-selling guide to HTML gives you the complete lowdown on all the changes and enhancements to the HTML, HTTP, and URL standards.

(BHTMLS) Our Price **\$26.95**





## Designing 3D Graphics

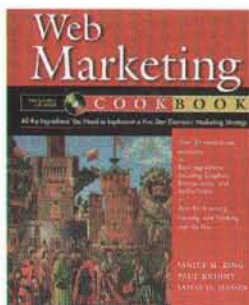
by Josh White

In this powerful book/CD-ROM package, top computer graphics artist Josh White tells you everything you need to know to create sophisticated real-time 3D graphics for computer games and virtual reality. This book contains the in-



depth knowledge of software tools and hands-on modeling techniques that Josh White has learned while creating artwork for over 20 commercial games, including Descent, Zone Raiders, Locus, Legoland, and others.

(BD3DG) Our Price **\$35.95**



## Web Marketing Cookbook

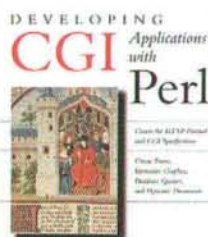
by Janice M. King, Paul Knight, and James H. Mason

Create the ultimate Web marketing site, quickly and painlessly! Learn how to build a Web site for your small business or nonprofit organization with this step-by-step



approach. This book/CD-ROM package contains your simple, non-technical tools for converting your print brochures, text, and graphics into a powerful online promotion.

(BWMCB) Our Price **\$35.95**



## Developing CGI Applications with Perl

by John Deep and Peter Holfelder

A complete step-by-step guide to creating sophisticated interactive CGI applications using Perl. If you're ready to build your own customized interactive documents, forms, graphics, and other full-feature CGI applications using Perl, then this book will show you

how. Covers CGI, HTTP, and the Perl scripting language.

(BDCGIA) Our Price **\$26.95**

## Rhapsody DeveloperOS Guide

by Jesse Feiler

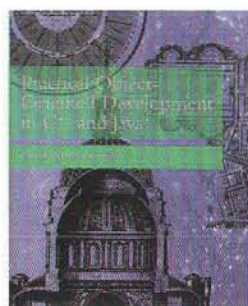
Covers the basic architectural principles of Rhapsody: the Mach microkernel, object-oriented programming, and the elements of a modern OS such as preemptive multitasking, protected memory, and symmetric multiprocessing. Also shows ways of getting to this new environment—Objective C, conversion tools, and the integration of Java—to develop Rhapsody products. Paperback, 450 pages.

(BRDG) Our Price **\$35.95**

**Check out our Web site!**

• Full product descriptions • Hundreds of more products

<http://www.devdepot.com>



## Practical Object-Oriented Development in C++ and Java

by Cay S. Horstmann

This book offers advice on real-world ways to use these powerful programming languages and techniques. Using the Unified Modeling Language (UML) methodology, expert

Cay S. Horstmann gives you clear, concise explanations of object-oriented design, C++, and Java in a way that makes these potentially daunting operations more accessible than they've ever been before.

(BPOOD) Our Price **\$31.50**

## WebMaster in a Nutshell, Deluxe Edition

by O'Reilly & Associates, Inc.

Cross-platform, completely portable, and lightning fast, the CD-ROM is an invaluable addition to the webmaster's toolbox. The CD-ROM contains the Web Developer's Library—the full text of the latest editions of five popular O'Reilly titles: "HTML: The Definitive Guide, 2nd Edition"; "JavaScript: The Definitive Guide, 2nd Edition"; "CGI Programming on the World Wide Web"; "Programming Perl, 2nd Edition"; and "WebMaster in a Nutshell." The Deluxe Edition also includes a printed copy of "WebMaster in a Nutshell," the all-inclusive quick reference that belongs next to every webmaster's terminal. Includes CD-ROM & 356 page book.

Requirements: The CD-ROM is readable on all platforms, but requires a web browser that supports HTML 3.2, Java, and JavaScript.

(BWMNUTD) Our Price **\$62.95**



## Programming For The Newton Using Macintosh, 2nd Edition

by Julie McKeehan and Neil Rhodes

This book gives you everything you need to create Newton 2.0 applications. From the people who developed the

Newton programming training materials for Apple Computer, this book uses a clear and comprehensive approach to teach you how to program the Newton. Includes a CD-ROM full of sample code and additional goodies. The samples include solutions to all of the coding examples found in this book. Each example and exercise also has a narrated QuickTime movie showing the solution from start to finish in Newton Toolkit.

(BPFNUM2) Our Price **\$31.45**







## Wireless For The Newton

by Julie McKeehan and Neil Rhodes



A book that picks up where Programming for the Newton left off, teaching the reader how to develop Newton software on the Macintosh. The enclosed floppy disk provides a sample application, as well as a fully functional

demonstration version of Newton Toolkit.

- Learn to develop Newton software on the Macintosh
- Hands-on Newton environment training with sample code
- Includes disk with sample source code for a Newton application, as well as demonstration NTK – the complete development environment for the Newton

(BWIRELESS) Our Price **\$31.45**



## JavaScript & Netscape Wizardry

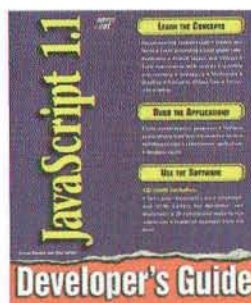
by Dan Shafer

The perfect book to show you how to turn Netscape into your own personal, customized operating system. Provides the inside tips and techniques for making your Web



pages much more attractive. Shows you how to use all of the key features of the JavaScript language, including objects, methods, properties, events, and much more. Includes CD-ROM with numerous interactive scripts written in JavaScript you can add to your Web pages today. A complete set of the best Java applets. Useful plug-ins designed to supercharge Netscape and resources to help JavaScript programmers.

(BJNWIZ) Our Price **\$31.45**



## JavaScript 1.1 Developer's Guide

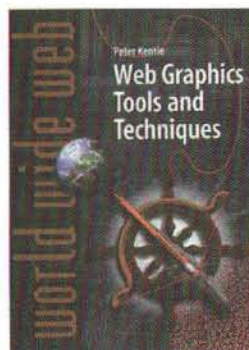
by Arman Danesh and Wes Tatters

Written by developers for developers. An advanced guide to creating professional Web applications with JavaScript 1.1 as deployed in Netscape Navigator 3.0, Microsoft Internet



Explorer 3.0, and LiveWire. Includes CD-ROM with Sun's Java Developer's Kit, JavaScript and HTML Editors for Windows and Macintosh, 20 contributed ready-to-run JavaScripts and JavaScript examples from the book.

(BJSDG) Our Price **\$44.99**

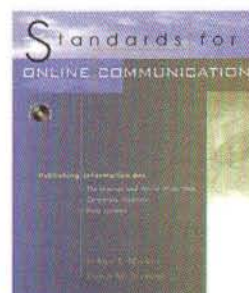


## Web Graphics Tools and Techniques

by Peter Kentie

The ultimate source of information on Web graphics for both Macintosh and PC users. Recent technologies covered include: ActiveX, Sound, Adobe Acrobat, Java, VRML, QuickTime VR and video, Shockwave, and 3D Animation.

(BWGTT) Our Price **\$35.95**



## Standards For Online Communication

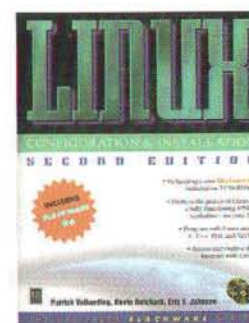
by JoAnn T. Hackos and Dawn M. Stevens

Guidelines for how to place information online within your company. Provides both a design and development process and



a set of guidelines for the Internet, intranets, and help systems for designers and authors who need to create effective electronic information. Includes CD-ROM with software containing files to help you utilize the models described in the book.

(BSFOC) Our Price **\$40.45**



## Linux Configuration & Installation, 2nd Edition

by Patrick Volkerking, Kevin Reichard, and Eric F. Johnson

Linux, the leading UNIX variant, has garnered loads of attention within the UNIX community. The amazing thing about

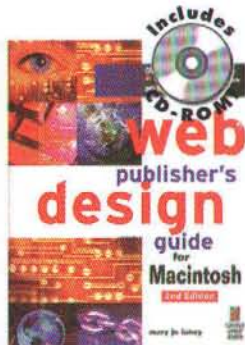


Linux is that you don't need a workstation to run it. Linux Configuration & Installation, Second Edition lets you run Linux today. Program with Linux using C, C++, Perl, and Tcl/Tk. The 2 CD-ROM pack offers one of the most popular Linux distributions, Slackware 96, and comes directly from Patrick Volkerking, the creator of Slackware.

(BLCI2) Our Price **\$35.95**







## Web Publisher's Design Guide for Macintosh, 2nd Edition

by Mary Jo Fahey

This is the only book that takes you step-by-step through real projects designed by talented new media artists. Internet design experts share their design secrets and art files (look for art files on the companion CD-ROM

by artist's last name). This expanded new edition includes Photoshop, Illustrator and DeBabelizer tricks from the first edition plus countless new ways to liven up your Web pages with 3D graphics, sound, movies, and much more.

(BWPDG2) Our Price **\$35.99**

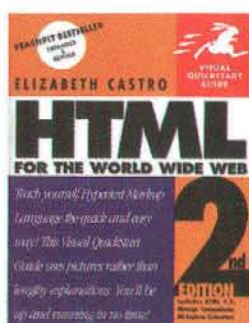
## BASIC for the Newton

by John Schettino and Liz O'Hara

Program on Macintosh, Windows-based PC, or on the Newton itself. Straight-forward "programming by example" approach - you'll be writing Newton programs right away. Includes 3.5" disk containing Demonstration NS BASIC and over fifty example programs (Newton not included).

(BNEWT) Our Price **\$32.35**

SEE RELATED CATEGORY: Dev. Environments

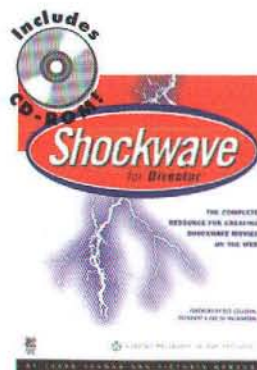


## HTML For The World Wide Web, 2nd Edition

by Elizabeth Castro

Teach yourself Hypertext Markup Language the quick and easy way! This Visual QuickStart Guide uses pictures rather than lengthy explanations. You'll be up and running in no time. If you need to learn HTML fast - this is book is for you.

(BHTMLW2) Our Price **\$16.15**



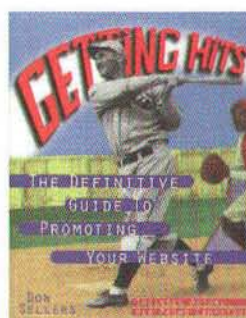
## Macromedia Shockwave for Director

by Jason Yeaman and Victoria Dawson

The complete resource for creating Shockwave movies on the Web. This hands-on reference makes it easy to create Shockwave movies and put them on the Web. Expert tips from the creators of Macromedia's first

Shockwave movies, together with detailed examples and instruction, provide everything you need to get started. Includes CD-ROM.

(BMSFD) Our Price **\$27**



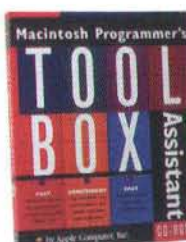
## Getting Hits-The Definitive Guide To Promoting Your Website

by Don Sellers

Getting Hits explains in easy-to-understand language the underlying concepts behind the art of Web site promotion. Just a few of the topics you'll learn include: using search

engines with URL's; finding related Internet groups or lists; understanding the nuances of click throughs and ad rates; and distributing press releases to key Internet contacts. With this book, you'll go beyond the conceptual and actually follow real-world tested promotional campaign strategies. Bring the world to your Web site!

(BGHITS) Our Price **\$17.95**



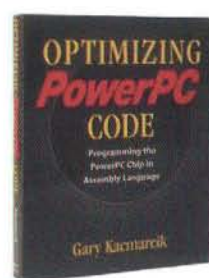
## Programmer's Toolbox Assistant CD-ROM

Instant electronic access to Inside Macintosh essentials. by Addison-Wesley Publishing

Get quick access to reference pages for over 4,000 Toolbox calls in your system software from their development environment. Essential

information for Macintosh software developers. Hypertext links allow programmers to view related topics easily. The ultimate electronic reference tool for Macintosh programmers.

(STBASST) Our Price **\$89.95**



## Optimizing PowerPC Code: Programming the PowerPC in Assembly Language

by Gary Kacmarcik

Take full advantage of the potential of the PowerPC by mastering the Assembly Language techniques. Learn to produce faster more robust software!

(BOPTPPC) Our Price **\$35.96**



## JavaScript For The World Wide Web

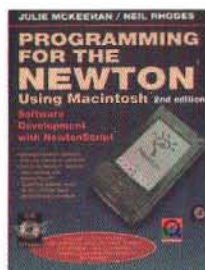
by Ted Gesing and Jeremy Schneider

This book takes an easy, visual approach to teaching JavaScript, where pictures guide you through the software and show you what to do. Works like a reference book, you look up what you need and then get straight

to work. No long winding passages, concise, straightforward commentary explains what you need to know.

(BJWWW) Our Price **\$16.15**





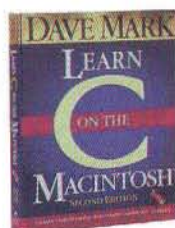
## Programming for the Newton Using Macintosh: Software Development with NewtonScript- Second Edition

by Julie McKeehan & Neil Rhodes



Praise for the Second Edition! "Rewritten from cover to cover, this new edition teaches you the essentials of programming for Newton 2.0. You must read this book. Then read it again. And again..."--SCRIBBLES (OxNUG, Australian Newton Users Group) Includes one CD-ROM for Macintosh 68030 or higher. 466 pp.

(BPFNUM) Our Price **\$31.45**



## Learn C on The Macintosh Second Edition

by Dave Mark



New revised edition! Easy-to-understand -- everything you need to start programming. Updated and enhanced exercises that lead you step by step.

You'll learn function, variables, point datatypes, data structures, file input and output and more! Includes CD-ROM with Metrowerks CodeWarrior™ Lite.

(BLEARNC2) Our Price **\$33.25**

SEE RELATED CATEGORY: Dev. Environments



## OBJECTIVE-C Object-Oriented Programming Techniques

by Lewis J. Pinson and Richard S. Wiener

Presents the basic concepts of object-oriented design and programming, and provides a precise description of the Objective-C

language. Several small-to-medium sized applications using Objective-C illustrate the general principles of object-oriented programming. Covers the two main versions of the Objective-C language. Demonstrates the versatility and power of the NeXT machine as a platform to support object-oriented programming. Shows how to design, implement, and use hierarchies of classes. Explains the purpose of pre-defined classes and shows how they can be used in designing programs.

(BOBJCOOPT) Our Price **\$34.15**

## Inside CodeWarrior Professional

by Metrowerks

Includes CodeWarrior IDE User's Guide. This is the printed version of the documentation provided on the CD. Covers CodeWarrior Professional Release, the debugger and associated tools.

(BINSWP) Our Price **\$34.95**

SEE RELATED CATEGORY: Dev. Environment

**Check out our Web site!**

• Full product descriptions • Hundreds of more products  
<http://www.devdepot.com>

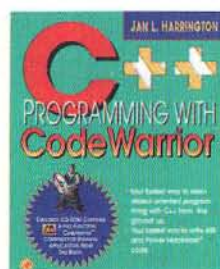
## Metrowerks CodeWarrior Programming

by Dan Parks Sydow



Includes CodeWarrior Lite, and Full Coverage of PowerPlant™. The best information on Metrowerks CodeWarrior, giving full coverage to the Gold Edition. CD includes Code Warrior Lite.

(BCWPROG) Our Price **\$35.95**



## C++ Programming with CodeWarrior

by Jan L. Harrington



Beginning OOP for the Macintosh and Power Macintosh and Mac OS compatibles. Learn object-oriented programming techniques using C++ as the example language and Metrowerks and CodeWarrior as the example

compiler. Enclosed CD contains example code from the book and a full-function Metrowerks CodeWarrior.

(BCPPCW) Our Price **\$32.35**

## CodeWarrior Software Development Using PowerPlant

by Jan L. Harrington



C++ programmers will learn to develop object-oriented software applications for the Mac and Power Mac using the PowerPlant environment and the classes that support it. Covers CodeWarrior 8. Included CD-ROM contains source code for all the programming examples in the book and Metrowerks CodeWarrior Lite.

(BCWSWDEV) Our Price **\$31.45**

## Inside PowerPlant

by Metrowerks

Create PowerPlant applications using the CodeWarrior IDE and PowerPlant Constructor. Full descriptions of major PowerPlant classes and resources. Included are the PowerPlant Constructor Manual, including View, TextTraits and Custom Types editing, and PowerPlant Library Reference, covering all classes and functions in PowerPlant.

(BINSPP) Our Price **\$34.95**

SEE RELATED CATEGORY: Dev. Environment

## AppleScript Finder Guide, English Dialect

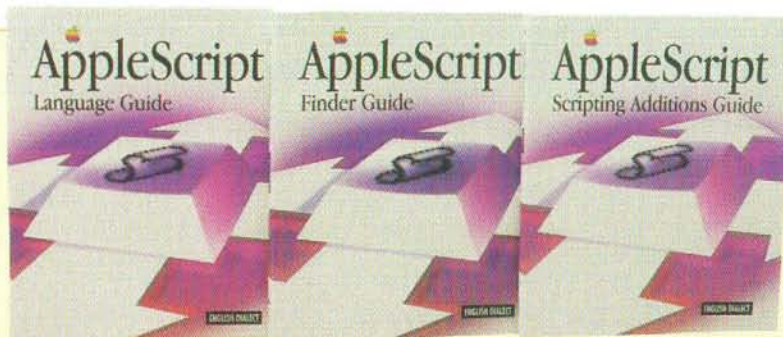
by Apple Computer, Inc.

Provides definitions for Finder object classes and commands. Write, record, or run scripts that trigger the same desktop actions that you trigger using the keyboard and mouse.

(BAFG) Our Price **\$17.95**

SEE RELATED CATEGORY: Scripting





## AppleScript Language Guide

by Apple Computer, Inc.

A complete reference for anyone using AppleScript to modify existing scripts or to write new ones. Contains useful information for programmers who are working on scriptable applications or complex scripts. Features detailed definitions of AppleScript terminology and syntax in the following categories: Value classes, commands, objects and references to objects, expressions, control statements, handlers, and script objects. Includes many sample scripts, discusses advanced topics such as writing command handlers for script applications, the scope of script variables and properties declared at different levels in a script, and inheritance and delegation among script objects.

(BALG) Our Price **\$26.95**

SEE RELATED CATEGORY: Scripting

## AppleScript Applications: Building Applications with FaceSpan and AppleScript

by John Schettino Affiliation & Liz O'Hara

Build complete AppleScript applications using FaceSpan, a user interface development tool that makes AppleScript applications truly "Mac-Like". Uses a step-by-step approach to demonstrate techniques for building applications through illustrations and samples. Provides Graphical User Interface (GUI) design tips and practical approaches for implementation. Contains one CD-Rom with AppleScript 1.1, a demonstrations version of FaceSpan 2.1, source code for all example applications numerous AppleScript shareware and demonstrations programs. Contains a section on debugging AppleScript applications using FaceSpan.

(BAPSCAP) Our Price **\$31.45**



## Special Edition Using CGI, 2nd Edition

by Jeffrey Dwight, Michael Erwin  
and Robert Niles

This complete reference provides professional Web developers and advanced personal users with the latest information

on using CGI (Common Gateway Interface) to interact with databases.

- Explains client and server uses of CGI
- Provides extensive coverage of live audio and video feeds, user chat and interaction, and CGI security
- Features separate chapters devoted to language-specific tips, tricks, and traps
- CD ROM is loaded with the HTML and CGI sample code from the book
- Includes applications for guest books, mail and new gateways, browser identification, access restriction, and shopping carts

(BSEUCGI) Our Price **\$44.99**



## Java in a Nutshell, 2nd Edition

by David Flanagan

A detailed overview of all of the new features in Java 1.1, both on a package-by-package basis and in terms of overall functionality. A comprehensive tutorial on "inner classes" that explains how to use all of the new types

of inner classes: static member classes, member classes, local classes, and anonymous classes. Practical, real-world example programs that demonstrate the new features in Java 1.1, including object serialization, the new AWT event handling model, internationalization, and a sample Java Bean.

(BJNUT2) Our Price **\$17.95**



## JavaScript for the Macintosh

by Matt Shobe and Tim Ritchey

Allows non-programmers to take advantage of the power of Netscape Navigator. Expand the capabilities of your Web page, without having to understand C or C++. CD-ROM contains "Wizlets" that allows you to easily create your own JavaScripts. Takes you step-by-step through programming cross-platform JavaScripts. Details how to create JavaScripts for JavaScript-aware Web browsers.

(BJAVASCRIPTJ) Our Price **\$40.50**





**Check out our Web site!**

• Full product descriptions • Hundreds of more products  
<http://www.devdepot.com>

## Teach Yourself Java for Macintosh in 21 Days

by Laura Lemay and Charles L. Perkins  
with Timothy Webster

Add interactivity and multimedia to Web pages! A step-by-step guide to make your Website come alive. Learn the basics of programming

Java applets and the concepts behind the Java language. Includes CD-ROM with a limited version of Roaster, the first commercial, integrated applet development environment for Java for the Macintosh!

(BJAVAMAC) Our Price **\$36**

## 3D Graphics Programming Using QuickDraw 3D

by Apple Computer, Inc.

Incorporate spectacular 3D graphics into your applications. Explore QuickDraw 3D, a revolutionary graphics extension to the Mac OS for Power Macintoshes. CD contains the complete QuickDraw 3D

system itself and a complete database of the QuickDraw 3D API, allowing you instant access to the hundreds of graphics calls via a fast viewing engine. Book/CD-ROM, 640 pages.

(B3DGRAP) Our Price **\$35.96**

## Tricks of The Mac Game Programming Gurus

by McCornack, Ragnemalm, Celestin, et al.

For beginning to expert game programmers. Complete overview of all the necessary components of game programming on the Macintosh. Packed with valuable tools,

utilities, sample code, CodeWarrior™ Lite and game demos. QuickDraw 3D and Power Mac optimization and inside info on how Glypha III was created. Hundreds of tried-and-true tricks, tips, and insider secrets from well-known Mac game programming experts.

(BTRICKS) Our Price **\$45**

## Black Art of Macintosh Game Programming

by Kevin Tieskoetter

Develop your own 3D games in C on the Mac. Includes CD with project files for both Symantec C and Code Warrior. Create freeform texture-mapped games and polygon graphics. Control dynamic source

code - all compatible as native to the Power Mac. Write directly to the screen, bypassing QuickDraw.

(BBLACK) Our Price **\$35.99**

## Advanced Color Imaging on the Mac OS

by Apple Computer, Inc.

Enhance your software's color capabilities with step-by-step instructions. Augment the color support supplied with QuickDraw, and QuickDraw GX. Use the Palette Manager to get the best colors on limited displays. Match

colors between screens and input/output devices (scanners & printers). CD includes a complete reference information in both QuickView and Acrobat formats. Plus, a sample application demonstrating ColorSync programming techniques.

(BADVCI) Our Price **\$33.25**

## The Internet Marketing Plan

by Kim M. Bayne

This book gives you a comprehensive framework for producing and executing a customized Internet marketing plan. Marketing communications veteran Kim Bayne supplies you with a clear set of step-by-step procedures for establishing, implementing, evaluating, and managing your company's online presence.

(BTIMP) Our Price **\$35.99**

## Protect Your Privacy on the Internet

by Bryan Pfaffenberger

This book/CD-ROM package gives you proven privacy defense strategies and techniques to help you make the Net a safer place to work and play. You'll get the names of Internet privacy organizations that are working to protect your privacy rights and find out what you can do to help.

(BPYP) Our Price **\$26.99**

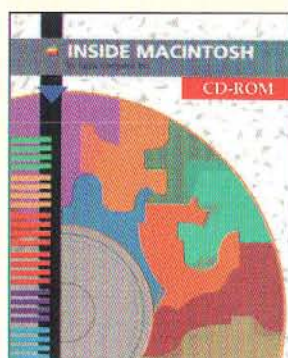
**Order Toll-free**  
**800-MACDEV-1**  
(800-622-3381)





# INSIDE MACINTOSH

by Apple Computer, Inc.



## Inside Macintosh: CD-ROM

by Apple Computer, Inc.

More than 25 volumes in electronic form.

Includes: QuickDraw™ GX Library, Macintosh Human Interface Guidelines, PowerPC System Software, Macintosh Toolbox Essentials and More Macintosh Toolbox, QuickTime and QuickTime Components. Access over 16,000 pages of information with Hypertext linking and extensive cross referencing.

(BIMCD) Our Price **\$89**

**WAIT...  
There's  
More!**

Limited time offer to buy these Inside Macintosh products – **10 % off!** For full product descriptions please see our Web site, or feel free to call, fax, or E-mail us.

PRODUCT	CODE	OUR PRICE
Inside Macintosh: Devices	BIMDEV	26.95
Inside Macintosh: Files	BIMFIL	26.95
Inside Macintosh: Imaging with QuickDraw	BIMIMAG	29.65
Inside Macintosh: Interapplication Communications	BIMIAPP	33.25
Inside Macintosh: Memory	BIMMEM	22.45
Inside Macintosh: More Macintosh Toolbox	BIMMAC	31.45
Inside Macintosh: Networking	BIMNET	26.95
Inside Macintosh: Operating System Utilities	BIMOPUSU	26.05
Inside Macintosh: Overview	BIMOVER	22.45
Inside Macintosh: PowerPC Numerics	BIMPPCNUM	26.05
Inside Macintosh: PowerPC System Software	BIMPPCSYS	22.45
Inside Macintosh: Processes	BIMPROC	20.65
Inside Macintosh: QuickDraw GX Environ. & Utilities	BIMGXENV	28.75
Inside Macintosh: QuickDraw GX Graphics	BIMGXGR	28.75
Inside Macintosh: QuickDraw GX Objects	BIMGXOBJ	28.75
Inside Macintosh: QuickDraw GX Printing	BIMGXPRNT	26.95
Inside Macintosh: QuickDraw GX Printing Extensions	BIMGXEXT	26.95
Inside Macintosh: QuickDraw GX Prog. Overview	BIMGXOV	22.45
Inside Macintosh: QuickDraw GX Typography	BIMGXTYP	26.95
Inside Macintosh: QuickTime	BIMQT	26.95
Inside Macintosh: QuickTime Components	BIMQTCOM	31.45
Inside Macintosh: Sound	BIMSOUND	26.95
Inside Macintosh: Text	BIMTEXT	35.95
Inside Macintosh: X-Reference	BIMXREF	17.95

(Book sale prices are contingent upon availability)



# MacTech®

M A G A Z I N E

## MacTech® Magazine

MacTech keeps Mac programmers & developers up to date with everything they need to know about software development. Topics like Rhapsody, Java, QuickTime, OPENSTEP, Objective-C, C/C++, Object Oriented Technologies, product reviews and much more!

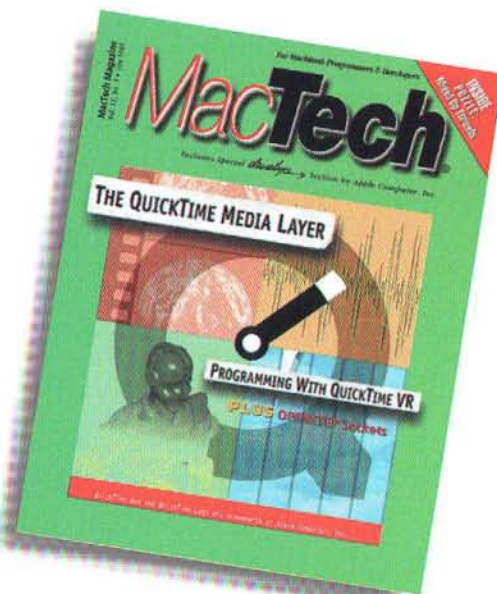
Subscriptions:

(MTYRDM) US/Domestic **\$47** for 12 issues

(MTYRCM) Canadian **\$59** for 12 issues

(MTYRFM) International **\$97** for 12 issues

**Back Issues: \$10** each plus shipping (subject to availability)



## MacTech® CD-ROM Volumes 1-12

- Includes Apple's *develop* issues 1-29 (1990-1997)
  - Almost 1600 articles from all 139 issues of MacTech Magazine (1984-1996) and through May of 1997
  - Improved hypertext, improved indices, and a new THINK Reference Viewer – for lightning quick access!
  - New hyperlinks between articles
  - 100+ MB of source code — use them in your applications, with no royalties!
  - Full version of THINK Reference™ — the original online guide to Inside Macintosh, Vols. I-VI
  - 80MB of FrameWorks/SFA archives and the most complete set of FrameWorks archives known
  - Sprocket™! MacTech's tiny framework that compiles quickly and supports System 7.5 features
  - The best threads from the Macintosh programmer newsgroups plus thousands of notes, tips, snippets, and gotchas
  - Popular tools that Macintosh programmers use to increase their productivity and much more!
- (SMTCD12) Volumes 1-12 Our Price **\$129**  
(SMTCD12U) Upgrade from any previous version Our Price **\$49**



**WAIT...  
There's  
More!**

Here are more products. For full product descriptions please see our Web site, or feel free to call, fax, or E-mail us. **Our prices on books are at least 10% off list price.**

PRODUCT	CODE	PRICE	PRODUCT	CODE	PRICE
<b>Development Environments</b>					
AppleScript Applications: Building Applications w/FaceSpan .....	BAPSCAP	\$31.45	Macintosh C Programming Primer Volume I .....	BCPRIM1	24.25
Basic for the Newton - Programming using NS BASIC .....	BNEWT	32.35	Macintosh C Programming Primer Volume II .....	BCPRIM2	24.25
C++ Programming .....	BCPPMACAP	31.46	Macintosh Pascal Programming Primer Volume I .....	BPASCPRI	24.25
C++ Programming with CodeWarrior .....	BCPPCW	32.35	Mastering the Think Class Library .....	BMASTERTCL	26.95
C/C++ SDK User's Guide .....	BCPPUSER	29.00	Mastering the Toolbox Using THINK C .....	BCPRIM2	24.25
CodeWarrior Inside PowerPlant .....	BINSPP	34.95	Metroworks CodeWarrior Programming .....	BCWPROG	35.95
CodeWarrior Software Development using PowerPlant .....	BCWSWDEV	31.45	Objective-C - Object-Oriented Programming .....	BOBJCOOPT	34.15
Cyberdog Programmers Kit .....	BCYBERDOG	34.95	Presenting Magic Cap .....	BPRESMAGIC	15.25
Dan Shafer Presents the Power of Prograph CPX .....	BDANPRO	19.95	Real World Apple Guide .....	BREALWLD	35.95
Inside CodeWarrior Book .....	BINSCW	34.95	Symantec C++ Programming .....	BSYMCPP	39.50
Instant CORBA .....	BIC	17.99	Taligent's Guide to Designing Programs .....	BTALIGENT	17.55
Last Resort Programmers Edition .....	BLSTRSRT	74.95	The Power of Prograph CPX .....	BDANPRO	19.95
Learn C on the Macintosh, 2nd Edition .....	BLEARINC2	33.25	Visual Programming with Prograph CPX .....	BVISPRO	30.60
Learn C++ on the Macintosh .....	BLRNCPP	35.05	Wireless For The Newton .....	BWIRELESS	31.45



## Hardware

Apple CD-ROM Handbook	BCDHAND	14.36
Designing Cards & Drivers for the Macintosh	BCARD	26.96
LaserWriter Reference	BLASERREF	17.96
PCI System Architecture 3rd Edition	BPCISYS	31.46
PowerPC System Architecture	BPPCARCH	31.46

## Internet Related

1994 Internet White Pages	B94WHITE	26.95
Active Java	BACTJAVA	23.36
America Online for Dummies	BAOLDUM	17.95
CGI By Example	BCGIBE	31.45
Building & Maintaining an Intranet w/ the Macintosh	BBAMAI	45.00
Claris Home Page Companion	BCHPC	26.95
Computer Privacy Handbook	BPRIV	22.45
E-Mail Essentials	BEMAIL	22.45
Elements of E-Mail Style	BEMAIL	13.45
Hooked on Java	BHJAVA	26.95
Instant Internet Guide	BINSTANT	13.45
Internet Book	BTHENET	22.50
Internet for Dummies 2nd Edition	BNETDUM2	17.99
Internet for Dummies Quick Reference	BDMQCK	8.05
Internet for Macs for Dummies	BNETDUM	17.95
Internet for Macs for Dummies Bestseller Edition	BIMFDBE	35.99
Internet Power Tools	BPWRTOL	36.00
Internet Publishing with Adobe Acrobat	BIPWAA	36.00
Internet, The, Deluxe Edition	BNETDELUX	31.50
Intranet Web Dev.: Enterprise Alternatives to Client/Server	BINTWD	44.99
Java Essentials for C/C++ Programmers	BJAVAESSEN	17.95
Java in a Nutshell	BJAVANUT	13.45
Java Language API SuperBible	BJLAS	53.99
Java Programming with CORBA	BJPWC	26.99
JavaScript for Macintosh	BJAVASCRPT	40.50
Learn HTML on the Macintosh	BLHTML	26.95
Learn Java on the Macintosh	BLJAVA	31.45
Mastering Netscape 2.0 for Macintosh, Second Edition	BMASNET2	36.00
More Internet for Dummies Starter Kit	BDMNET	17.95
Mosaic for Dummies	BMOSDUM	17.99
Net Chat	BNETCHAT	17.00
NetObjects Fusion Handbook	BNETOFH	45.00
Netscape Navigator 3.0	BNETN3	26.96
Netscape Navigator Starter Kit for Macintosh	BNETNSK	31.49
Perl Quick Reference	BPERLREF	17.99
Planning and Managing Websites	BPLANWEB	35.96
Protect Your Privacy on the Internet	BPYP	26.99
Providing Internet Services via the MacOS	BPROVNET	31.46
Publish it on the Web	BWEBPUB	31.46
TCP/IP Vol 1-Vol 2 Bundle	BTCP12BNDL	99.00
Teach Yourself Java in 21 days	BJAVAMAC	36.00
The Internet Marketing Plan	BTIMP	35.99
Underground Guide to Telecommuting	BUNDER	22.45
Using Lotus Notes as an Intranet	BULN	40.45
Web Heard Mac Guide	BWEBHEAD	22.45
Web Page Scripting Techniques	BWEBPST	45.00
Web Publishing with Adobe Acrobat and PDF	BWPWAA	35.95
Web Weaving	BWWEAV	22.45
Webmaster Macintosh	BWEBMAS	26.95

## Scripting and Solutions

AppleScript Applications: Building Apps with FaceSpan	BAPSCAP	31.45
AppleScript Scripting Additions Guide	BSCRADD	17.05
Applied Macintosh Scripting	BAPPLIED	31.45
Complete HyperCard 2.2 Handbook	BHYPCRD2	31.50
Complete HyperTalk 2.2	BHYPCRD2	31.50
Danny Goodman's Apple Guide Starter Kit	BGGAGSK	31.46
HyperCard Stack Design	BHYPSTA	19.95
JavaScript for Macintosh	BJAVASCRPT	40.50
Perl Quick Reference	BPERLREF	17.99
Real World Apple Guide	BREALWLD	35.95

## Technical Reference

Active Java	BACTJAVA	23.36
Apple CD-ROM Handbook	BCDHAND	14.36
Art of Human Interface Design	BAIID	29.65
Black Art of Mac Game Programming	BBLACK	35.99
C++ for Dummies	BCPPDUM	17.95
C for Dummies Vol. 1	BCDUM	17.95
Developing Object Oriented Software for the Macintosh	BDEVOB	26.06
Extending the Mac toolbox	BETMT	22.46
Foundations of Macintosh Programming	BFOUND	35.96
Fragment of Your Imagination	BFRAG	35.96
Guide to Macintosh Software Localization	BLOCALIZ	24.26
Guide to Macintosh System 7.5	BSYS7.5	22.50

Inside AppleTalk	BAPTALK	31.45
Inside the Macintosh Communications Toolbox	BCOMM	22.45
LaserWriter Reference	BLASERREF	17.96
Learn C++ on the Macintosh	BLRNCPP	35.05
Learn C on the Macintosh, 1st Edition	BLEARNC1	31.45
Learn C on the Macintosh, 2nd Edition	BLEARNC2	33.25
Mac Programming for Dummies	BMACDUM	17.95
Macintosh C Programmer Primer Volume 1	BCPRIM1	24.25
Macintosh C Programmer Primer Volume 2	BCPRIM2	24.25
Macintosh OLE2 Prog. Reference Working with Objects	BOLE2	40.45
Macintosh Pascal Programming Primer Volume 1	BPASCPRI	24.25
Macintosh Programming Secrets 2nd edition	BPSECRET	28.76
Macintosh Programming Techniques	BPTCH	31.95
Microsoft Visual J++ 1.1 Sourcebook	BMVJS	35.95
More Mac Programming Techniques	BMORETECH	34.50
Network Frontiers Bundle	BNETFB	59.95
Newton Programming Guide	BNEWTPGUID	40.46
Object Oriented Programming Design	BOOPRODES	20.66
Optimizing PowerPC Code	BOPTPPC	35.96
Perl Quick Reference	BPERLREF	17.99
PostScript Language Reference	BPSLANREF	29.66
Powerbook: Digital Nomad's Guide	BPBTDNG	22.46
PowerPC Programmer's Toolkit	SPPCPT	40.50
Programming Introduction to the Macintosh Family	BFAMILY	22.46
Programming for System 7	BSYS7	24.25
Programming Primer Macintosh Volume 1	BPRIMMAC	34.15
Programming Starter Kit	BPROSTART	40.50
Programming with AppleTalk	BPROAT	22.45
QuickTime - Official Guide for Macintosh Users	BQTGUIDE	45.00
Real World Apple Guide	BREALWLD	35.95
ResEdit All Night Diner	BRESIDINE	22.45
ResEdit Complete, 2nd Edition	BRESID2	31.45
ResEdit Reference	BRESIDREF	26.96
Software by Design: Creating User Friendly Software	BDESIGN	26.95
Symantec C++ for the Macintosh: The Basics	BSCFTMTB	31.46
Teach Yourself Macintosh C++ in 21 Days	BCPP21D	26.99
Technical Introduction to the Macintosh Family	BTITTMF	24.26
Tog on Software Design	BTOG	26.95
Wireless for the Newton Development for Mobil Comm.	BWIRELESS	31.45
Writing Localizable Software	BLOCAL	24.25

## Miscellaneous

3D Graphics-Tips, Tricks, & Techniques	B3DGTIT	31.46
A Fragment of Your Imagination	BFRAG	35.96
Adobe Premiere for the Macintosh	BPREM	44.95
America Online for Dummies	BAOLDUM	17.95
AppleGuide Complete	BAPLGD	35.96
Art of Human Interface Design	BAIID	29.65
CD-ROM Guide to Multimedia Authoring	BCDMULTI	40.45
CompuServe for Dummies	BCSDUM	17.95
Creating Interactive CD-ROM	BINTERCDR	35.95
Cyberpunk Handbook	BCYBPUNK	8.95
Danny Goodman's Apple Guide Starter Kit	BGGAGSK	31.46
Danny Goodman's Macintosh Handbook	BGOOCHB	26.95
FrameWorks Source Code Disk	MTFWSC	9.95
FrameWorks Magazine Back Issue	MTFWBACK	8.00
Global Interface Design	BGLOBAL	32.35
Graphic Gems 2	BGEMS2	44.95
Graphic Gems 4	BGEMS4	44.95
Graphic Gems V	BGEMS5	44.95
Infini-D Revealed	BINFDRV	40.50
Inside Director 5 with Lingo for Macintosh	BID5WLM	44.99
Late Night with MacHack	BLATE	26.95
Mac Bathroom Reader	BBATH	11.70
Macromedia Director Lingo Workshop, 2nd Edition	BMDLW2	40.50
Mac Screamer: The Ultimate Macintosh Supercharging Kit	BSCREAM	31.50
Macintosh Crash Course	BCRASH	26.95
MacTech Back Issues	MTBACKISS	10.00
Macworld Ultimate Macintosh Programming Book	BULTMAC	35.95
Managing AppleShare & Workgroup Servers	BMAWS	26.95
MADACON '93 CD-ROM	SMADA93	9.95
Multimedia Authoring: Building and Developing Documents	BMMAUTH	31.45
Multimedia Starter Kit for Macintosh	BMMSTART	27.00
Network Frontiers Bundle	BNETFB	59.95
Profit from Experience	BPROFIT	22.45
ResEdit Complete, Second Edition	BRESID2	31.45
Sad Macs, Bombs and Disasters	BSADMAC	22.45
Stupid Mac Tricks	BSTUPIDMAC	17.95
The Elements of E-Mail Style	BEMAIL	13.45
The Software Developer's & Marketer's Legal Companion	BSDAMLC	33.26
Tog on Software Design	BTOG	26.95
Tricks of the Mac Game Gurus	BTRICKS	45.00
Zen and the Art of Resource Editing	BZAAORE	27.00



## —A—

<b>1000 Games for Macintosh</b> .....	17
<i>3D Graphics Programming Using QuickDraw 3D</i> .....	27
A.D. Software .....	12
Absoft Corporation .....	4
<b>Abuse</b> .....	17
<b>Additional Development Enviroments Listings</b> .....	5
<b>Additional Listings for Games</b> .....	17
<b>Additional Listings for Tools, Libraries &amp; Utilities</b> .....	11
Adianta Inc. ....	6
<i>Advanced Color Imaging on the Mac OS</i> .....	27
<b>AG Author</b> .....	11
AG Group, Inc. ....	10
Altura Software, Inc. ....	5
Apple Computer, Inc. ....	3, 14, 15, 18, 26, 28
<b>Apple Dylan Technology Release</b> .....	3
<b>Apple Guide Integration</b> .....	18
<b>Apple Media Tool Programming Environment 2.1</b> .....	15
<i>AppleScript Applications: Building Apps w/ FaceSpan &amp; AppleScript</i> .....	26
<i>AppleScript Finder Guide, English Dialect</i> .....	25
<i>AppleScript Language Guide</i> .....	26
<b>AppleScript Software Development Toolkit 1.1</b> .....	14
<b>AppMaker</b> .....	8
<b>Apprentice 6</b> .....	7
<b>AppSketcher 1.0 for BeOS</b> .....	7
<b>Audio Track</b> .....	16

## —B—

<b>B-Tree HELPER 2.2</b> .....	8
Bare Bones Software .....	8, 9, 12
<i>BASIC for the Newton</i> .....	24
<b>BBEdit 4.0.4</b> .....	9, 12
BeachWare, Inc. ....	11, 13, 16, 17
BeatWare, Inc. ....	7
<b>Bee-one</b> .....	11
<i>Black Art of Macintosh Game Programming</i> .....	27
Bowers Development .....	8
Bungie Software .....	17

## —C—

<i>C++ Programming with CodeWarrior</i> .....	25
<b>c-tree Plus® Database Handler</b> .....	6
Capilano Computing .....	10
<b>Captivate 4.6: Essential Graphics Utilities</b> .....	16
<b>Casino!</b> .....	17
Celestin Company .....	7
<b>CGI Toolkit</b> .....	13
<b>Classic Arcade</b> .....	17
<b>Clip VR™</b> .....	15
<b>CodeBuilder</b> .....	5, 9
<b>CodeManager 4</b> .....	2
<b>CodeWarrior for BeOS 3</b> .....	2
<b>CodeWarrior for PalmPilot</b> .....	2
<b>CodeWarrior Latitude</b> .....	2
<b>CodeWarrior Professional Release</b> .....	2
<i>CodeWarrior Software Development Using PowerPlant</i> .....	25
<b>CodeWarrior Wear</b> .....	19
<b>CompileIt!</b> .....	10
Conix Graphics .....	10
<b>CPU Doubler</b> .....	10

## —D—

<i>D'OLE Dev Guide</i> .....	20
<i>Debugging Macintosh Software with MacsBug</i> .....	21
<i>Designing 3D Graphics</i> .....	22
<b>DesignWorks 4.0</b> .....	10
<i>Developing CGI Applications with Perl</i> .....	22
Digital Technology International .....	14
Digitool, Inc. ....	4

<b>Discover Programming for Macintosh</b> .....	3
<b>Discover Programming with Java</b> .....	3
<i>Discovering OPENSTEP, Mac</i> .....	20
<i>Discovering OPENSTEP, Windows</i> .....	20
<b>dtF</b> .....	8
dtF Americas .....	8

## —E—

<i>EOF Developer's Guide for EOF 1.x</i> .....	20
<i>EOF Developer's Guide for EOF 2.0</i> .....	20
<i>EOF Developer's Guide for EOF 2.1 (Mac &amp; Windows)</i> .....	20
<b>EtherPeek</b> .....	10
eVox Productions .....	15
Excel Software .....	7

## —F—

<b>FaceSpan v2.1</b> .....	14
Faircom .....	6
<b>Future Basic II</b> .....	8

## —G—

<i>Getting Hits—The Definitive Guide To Promoting Your Website</i> .....	24
<i>Getting Started With WebObjects</i> .....	20
<b>Guide Composer™ 1.2</b> .....	6

## —H—

<i>HTML For The World Wide Web, 2nd Edition</i> .....	24
<i>HTML Sourcebook, 3rd Edition</i> .....	21

## —I—

<i>Increasing Hits and Selling More on your Web Site</i> .....	21
<i>Inside CodeWarrior Professional</i> .....	25
<i>Inside Macintosh: CD-ROM</i> .....	28
<i>Inside Macintosh: Devices</i> .....	28
<i>Inside Macintosh: Files</i> .....	28
<i>Inside Macintosh: Imaging with QuickDraw</i> .....	28
<i>Inside Macintosh: Interapplication Communications</i> .....	28
<i>Inside Macintosh: Memory</i> .....	28
<i>Inside Macintosh: More Macintosh Toolbox</i> .....	28
<i>Inside Macintosh: Networking</i> .....	28
<i>Inside Macintosh: Operating System Utilities</i> .....	28
<i>Inside Macintosh: Overview</i> .....	28
<i>Inside Macintosh: PowerPC Numerics</i> .....	28
<i>Inside Macintosh: PowerPC System Software</i> .....	28
<i>Inside Macintosh: Processes</i> .....	28
<i>Inside Macintosh: QuickDraw GX Environ. &amp; Utilities</i> .....	28
<i>Inside Macintosh: QuickDraw GX Graphics</i> .....	28
<i>Inside Macintosh: QuickDraw GX Objects</i> .....	28
<i>Inside Macintosh: QuickDraw GX Printing</i> .....	28
<i>Inside Macintosh: QuickDraw GX Printing Extensions</i> .....	28
<i>Inside Macintosh: QuickDraw GX Prog. Overview</i> .....	28
<i>Inside Macintosh: QuickDraw GX Typography</i> .....	28
<i>Inside Macintosh: QuickTime</i> .....	28
<i>Inside Macintosh: QuickTime Components</i> .....	28
<i>Inside Macintosh: Sound</i> .....	28
<i>Inside Macintosh: Text</i> .....	28
<i>Inside Macintosh: X-Reference</i> .....	28
<i>Inside PowerPlant</i> .....	25

## —J—

<i>Java in a Nutshell, 2nd Edition</i> .....	26
<i>JavaScript 1.1 Developer's Guide</i> .....	23
<i>JavaScript Cookbook</i> .....	21
<i>JavaScript for the Macintosh</i> .....	26
<i>JavaScript For The World Wide Web</i> .....	24
<i>JavaScript &amp; Netscape Wizardry</i> .....	23

## —L—

Lakewood Software .....	11
-------------------------	----



Late Night Software Ltd.	14
Learn C on The Macintosh Second Edition	25
Legtop Pdeum	19
Linux Configuration & Installation, 2nd Edition	23

## —M—

MacA&D 6.0	7
Macintosh Common Lisp 4.0	4
Macromedia Shockwave for Director	24
MacTech® CD ROM Vol. 1-12	29
MacTech® Magazine	18, 29
MacTech® Mouse Pad	19
Magreable Software	8
Main Event Software	14
Mainstay	4, 9, 12, 16
Mango Tree Software	14
Measuring the Impact of Your Web Site	21
Media Cleaner Pro	16
Memory Mine	6
Metrowerks	2, 3, 19
Metrowerks CodeWarrior Programming	25
MkLinux: Microkernel Linux for the Power Macintosh	4
Multimedia Authoring with Apple Media Tool	15
MultiWare Multimedia Collection	16
Music Tracks	16

## —N—

Newton Toolkit 1.6 (Mac/Win)	3
NeXT Software Inc.	20
Nisus Software	11
NS BASIC 3.6 for the Newton with Visual Designer	5
NS BASIC Corporation	5

## —O—

Object-Oriented Programming and Objective C	20
OBJECTIVE-C Object-Oriented Programming Techniques	25
ObjectMaster Professional Edition	5
ObjectSet Mail SDK	6, 12
Ornyx Technology	7
OOFIE Reporter Writer	12
OpenGL for the Macintosh	10
Optimizing PowerPC Code: Prog. the PowerPC in Assembly Language	24
Orchard Software	10

## —P—

PageCharmer 1.0	12
Phyla™: Object-Oriented Database	9
Pictorius Inc.	13
Power Box	11
Power MachTen	13
Practical Object-Oriented Development in C++ and Java	22
PreFab Player	14
PreFab Software, Inc.	14
Prime Time Freeware	4
Pro Fortran	4
Programmer's Toolbox Assistant CD-ROM	24
Programming for the Newton Using Macintosh	25
Programming For The Newton Using Macintosh, 2nd Edition	22
Purity Software, Inc.	12

## —Q—

QC7	
QUED/M 3.0	11
QuickTime Developer's Kit 2.0	15
QuickTime VR 2.0 Authoring Tools Suite	15

## —R—

r-tree Report Generator	6
Rach Inc.	19

Rhapsody DeveloperOS Guide	22
Roaster	5, 13
Roaster Technologies, Inc.	5, 13
Royal Software, Inc.	6, 10, 13, 14

## —S—

Script Debugger	14
ScriptDemon	6, 13
Scripter 2.0	14
ScriptGen Pro	8
Seapine Software, Inc.	9
Smartcode Software	6, 12
SoftPolish CD-ROM	8
Special Edition Using CGI, 2nd Edition	26
Spellswell Plus 2.1	6
Standards For Online Communication	23
Staz Software	8
Step-Up Installer Pack	8
StepUp Software	6, 8
StoneTable 68K/PPC	7
StoneTable Publishing	7
System 7.5 Technologies	18

## —T—

TCP/IP Scripting Addition	14
Teach Yourself Java for Macintosh in 21 Days	27
Tenon Intersystems	5, 9, 12, 13
Terran Interactive	16
TestTrack-Bug Tracking the Macintosh Way	9
The Internet Strategic Plan	21
The Marathon Trilogy Box Set	17
The Way Computer Graphics Works	21
The Web Navigator	21
Tools Plus libraries + framework	9
Tricks of The Mac Game Programming Gurus	27
Trivia Warehouse 2000	17

## —U—

UNI SOFTWARE PLUS	7
Using EOF 2.1 w/ OPENSTEP (Mac & Windows)	20

## —V—

VIP-BASIC: Visual Interactive Programming in BASIC	4
VIP-C: Visual Interactive Programming in C	4
Virtual Reality Programming with QuickTime VR 2.0	15
Virtual Tutor for QuickTime VR	18
Vivistar	10
VOODOO 1.8	7
VText	10

## —W—

Water's Edge Software	9
WAVES	16
Web Graphics Tools and Techniques	23
Web Marketing Cookbook	22
Web Publisher's Design Guide for Macintosh, 2nd Edition	24
Web Ware	11, 13
WebMaster in a Nutshell, Deluxe Edition	22
WebObjects Developer's Guide	20
WebSiphon	12
WebTen	12
WindowScript	14
Wireless For The Newton	23
Working Software	6
Working w/ Interface Builder (for EOF)	20

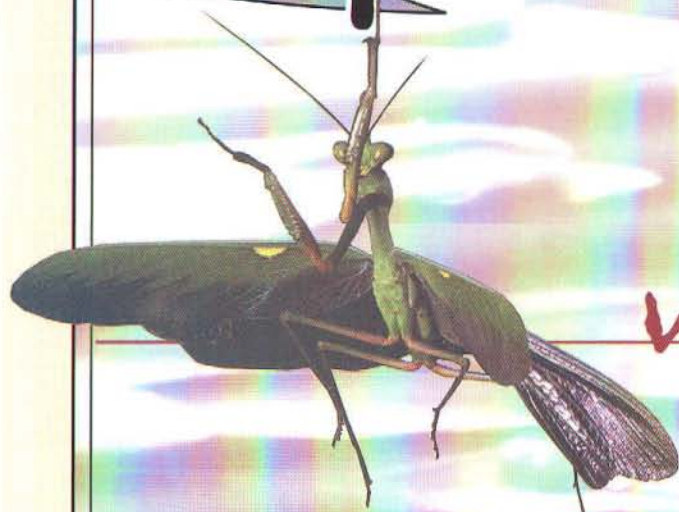
## —X—

XplainCorporation	18, 19, 29
-------------------	------------



# purity

software, inc.



## websiphon™

\$495

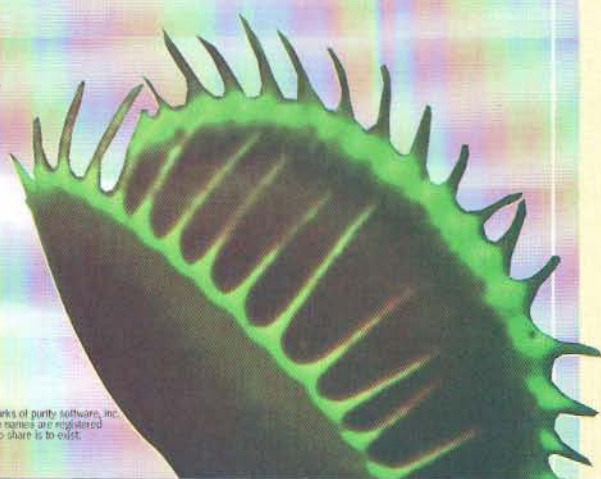
WebSiphon allows you to rapidly design custom server-side web applications by embedding an easy to learn, yet powerful scripting language directly into your HTML documents. WebSiphon also includes Verona, the only Macintosh flat-file database server written specifically for use on a web site. If you have ever felt restricted by other web site development products on the market, then this is the tool for you.

- PowerPC™ Native
- Multi-threaded
- Compiled Language
- Caching

## webSentinel™

\$99

WebSentinel replaces or supplements your web server's built-in security, allowing more flexibility and power to manage users and secure areas with a complete Macintosh user interface. Features include GREP match strings, custom "No Access" files for each realm, and full caching. WebSentinel's plug-in architecture allows you to integrate with your existing user data and security architectures.



demons

[www.purity.com](http://www.purity.com)  
1016 mopac circle, suite 101  
austin tx 78746  
phone 512.328.2288  
fax 512.328.2688  
[info@purity.com](mailto:info@purity.com)

purity software, websiphon, and websentinel are registered trademarks of purity software, inc. and copyright 1997. all rights reserved worldwide. all other trade names are registered trademarks of their respective holders. puro code for puro minus: to share is to exist.



# CODEWARRIOR

## LATITUDE

**CodeWarrior**  
Latitude

**KICKING  
BU TT  
AND  
WRITING  
CODE**

When Rhapsody gets here,  
you'll be ready...with  
CodeWarrior Latitude.™

It's Metrowerks' newest  
porting tool, designed to  
give you a leg up on  
Rhapsody. Recompile your  
Mac OS source code, link it  
with the Latitude libraries  
and find out which portions  
of your code are going to  
port smoothly...and which  
won't [forewarned is fore-  
armed]. As Rhapsody evolves,  
so will Latitude; registered  
users will receive all  
developer releases, the  
first full release, plus  
one additional update.  
CodeWarrior Latitude. \$399.  
The tools you need...and  
a little attitude to boot.

COOL TOOLS FOR KILLER CODE

